Universidad Zaragoza
1542

Trabajo Fin de Grado

Implementación de un sistema de comunicaciones por luz visible (VLC) basado en FPGA

Implementation of a visible light communication system (VLC) based on FPGA

Autor

Ángel Molina Egea

Director

Isidro Urriza Parroqué

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2019

Implementación de un sistema de comunicaciones por luz visible (VLC) basado en FPGA

RESUMEN

El crecimiento gradual del uso de LEDs para iluminación en detrimento de las tecnologías de alumbrado tradicionales ha abierto un nuevo campo de investigación en el que las comunicaciones VLC podrían ser una solución ante la escasez del espectro electromagnético disponible. En este trabajo se abordará el desarrollo de las partes analógica y digital básicas necesarias para la implementación de un sistema de comunicaciones por luz visible. Se diseñará en VHDL tanto la etapa de emisión como la de recepción del sistema correspondientes a la parte digital del mismo mediante la utilización de bloques IP, mientras que la parte analógica se montará en primer lugar en una placa de pruebas para, una vez verificado el funcionamiento, desarrollar el sistema en una PCB.

Implementation of a visible light communication system (VLC) based on FPGA

ABSTRACT

The gradual growth in the use of LEDs for lighting to the detriment of traditional lighting technologies has opened a new field of research in which VLC could be a solution to the scarcity of the available electromagnetic spectrum. This paper will address the development of the basic analog and digital parts, which are necessary for the implementation of a visible light communications system. Both the emission and reception stages of the system corresponding to the digital part will be designed in VHDL with the use of IP blocks, while the analog part will be mounted firstly on a protoboard in order to develop the system on a printed circuit board once the operation has been verified.

Índice

Capítulo 1

Introducción

1.1. Alcance y objetivos

En este Trabajo Fin de Grado se pretende mostrar el proceso de diseño e implementación necesario en el desarrollo de un sistema VLC (Visible Light Communication), tanto desde un punto de vista analógico como digital. El objetivo es encontrar una solución sencilla y funcional desde un punto de vista analógico, pudiéndonos centrar de este modo en el desarrollo del sistema digital, todo ello partiendo desde cero. El diseño tratará de un enlace unidireccional punto a punto en el que mediante un LED (Light Emitting Diode) enfrentado con un fotodiodo trataremos de enviar y recibir caracteres.

Para ello se hará uso de la modulación OOK (On-Off Keying) codificada sobre Manchester. Todo el sistema digital se desarrollará sobre una FPGA (Field Programmable Gate Array) en VHDL (VHSIC Hardware Description Language), utilizando para algunas funciones el uso de bloques IP (Intellectual Property). En nuestro caso, a diferencia de las actuales investigaciones sobre VLC, no se buscará la transmisión a grandes velocidades, sino un diseño sencillo y eficiente.

1.2. Planteamiento del trabajo

La primera tarea necesaria para el desarrollo del proyecto consiste en la investigación dentro de las comunicaciones VLC, con el fin de intentar hacernos una primera idea sobre estas conociendo algunas de sus características. Esto se puede llevar a cabo mediante la lectura de otros Trabajos Fin de Grado u otro tipo de artículos referentes al tema.

A continuación se deberá realizar un análisis desde el punto de vista analógico teniendo como objetivo principal la caracterización de los componentes necesarios tanto para recibir como para mandar caracteres.

Una vez llegados a esté punto, se puede comenzar con la implementación del sistema desde ambos puntos, analógico y digital. Desarrollando en primer lugar el montaje del circuito en una placa de pruebas, para más adelante, una vez todos los componentes estén seleccionados de forma correcta, diseñar todo en una PCB (Printed Circuit Board). Para la implementación de la parte digital se hará uso de la herramienta *Vivado Design Suite 2018.2.2*. El código de VHDL correspondiente, así como el fichero .xdc que configura la FPGA, junto con algún script de MATLAB utilizado para la realización de pruebas se encuentran disponibles públicamente en `https://github.com/AMolinaEgea/VisibleLightCommunication`.

1.3. Estructura de la memoria

Respecto a la estructura de la memoria, en el capítulo 2 se hace una breve introducción sobre las comunicaciones por luz visible, su grado de desarrollo y algunas aplicaciones potenciales, además de otras en las que ya se están usando. Seguidamente a lo largo del capítulo 3, se muestra el sistema desarrollado por nosotros para llevar a cabo las comunicaciones mediante luz visible, detallando cada bloque por separado haciendo una división entre la parte correspondiente al emisor y la del receptor. A su vez estás se subdividirán en parte analógica y digital. Por último, para finalizar el capítulo se aportará una breve descripción del canal. Posteriormente en el capítulo 4, se muestran las pruebas llevadas a cabo con el sistema en funcionamiento, mostrando además los pasos seguidos y las diferentes comprobaciones que se han ido realizando hasta conseguir el fin mayor. Por último, se elaboran las conclusiones obtenidas del desarrollo del proyecto.

Capítulo 2

Comunicaciones por luz visible

En este capítulo se va a tratar de introducir lo que son las comunicaciones por luz visible, su grado de desarrollo a día de hoy y por último, se enumerarán algunas de las tantas aplicaciones en las que se podría usar y que ya se está usando este tipo de tecnología.

2.1. Introducción

En la actualidad las comunicaciones por luz visible han experimentado un giro notable debido principalmente a la sustitución gradual de las fuentes de luz tradicionales por LEDs. Esto es debido en gran medida a una serie de características tales como: un menor coste respecto a las anteriores fuentes de luz, gran potencia de emisión, una alta eficiencia y un tiempo de vida duradero. Con esto se ha abierto un inmenso abanico de posibilidades abordables mediante LEDs, convirtiéndose en la fuente de luz por excelencia para VLC [1]. Todo esto convierte este tipo de comunicaciones en un tema de gran interés para su estudio.

Este concepto conocido como VLC y el cual se aborda en este Trabajo Fin de Grado se basa en la transmisión de información mediante LEDs en el espectro de la luz visible. Esto se consigue a través de la modulación de la intensidad de la luz a velocidades más rápidas que la respuesta del ojo humano, obteniendo así transmisión de información a la vez que iluminación.

Como en todo sistema de comunicaciones, partiremos de la idea más básica donde como factor común tenemos 3 bloques: la fuente, la cual será la encargada de mandar el mensaje, el canal, lugar por donde transmitiremos nuestro mensaje, en VLC el aire, y por último el receptor, que se encargará de recibir el mensaje proveniente de la fuente.

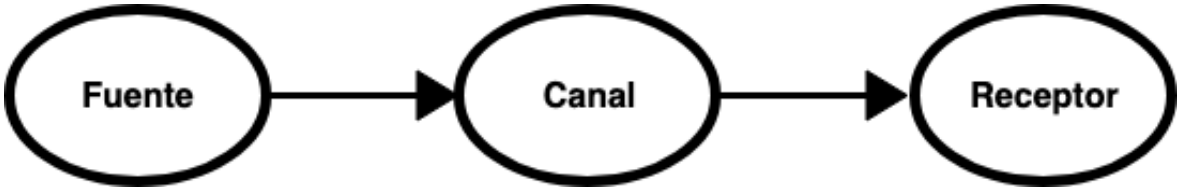

Figura 2.1: Diagrama de bloques de un sistema VLC

A lo largo del trabajo se irán explicando las diferentes opciones seleccionadas para la constitución de los bloques de la figura anterior, además de algunas alternativas que se podrían haber tomado.

Por último y con el fin de obtener una idea más concreta de VLC, se muestra a continuación una imagen de la zona del espectro electromagnético en la que se realizan este tipo de comunicaciones, haciéndonos así una idea del rango de de longitudes de onda abordado:

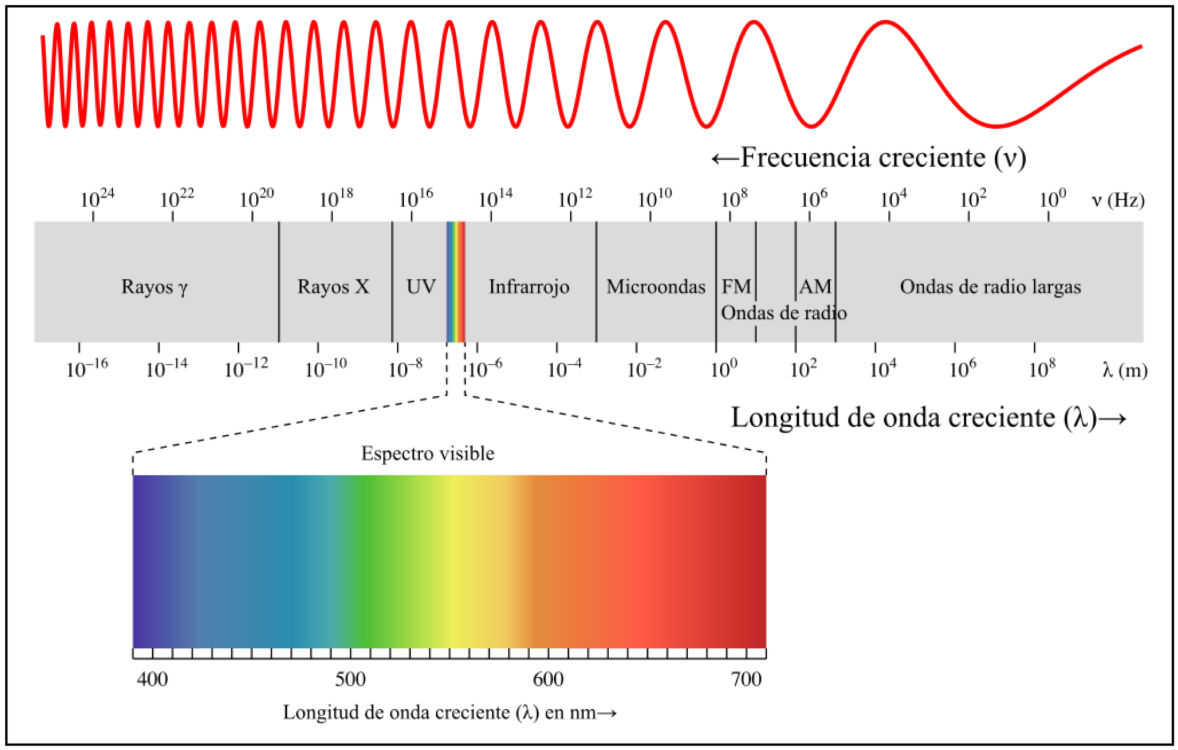

Figura 2.2: Espectro visible

Como se puede observar en la imagen, el espectro donde vamos a trabajar al montar este tipo de sistema de comunicaciones irá aproximadamente desde los 400 nm a los 790 nm.

2.2. Estado del arte

Se podría considerar que el primer experimento de comunicaciones por luz visible fue el fotófono, inventado por Alexander Graham Bell en 1880. Mediante este instrumento se consiguió la transmisión del sonido a través de emisiones de luz. Debido a la enorme cantidad de problemas que presentaban este tipo de comunicaciones, en los siguientes años fueron eclipsadas en gran parte por las radiocomunicaciones.

El origen del término VLC como se conoce en la actualidad, empezó a coger forma por el año 2000 en Japón. Debido al interés que levantaba este tipo de sistema de comunicaciones, rápidamente fue extendido al resto del mundo. En 2003 con el fin de encontrar nuevas aplicaciones para VLC se estableció el VLCC (Visible Light Communication Consortium). El desarrollo de este tipo de comunicaciones ha sido tal que en 2016 se llegó a conseguir en un sistema MIMO (Multiple-Input Multiple-Output), mediante un LED comercial de luz blanca y un fotodiodo PIN, velocidades por encima de 1 Gb/s a una distancia de 0.6 m [2], y velocidades de 2.08 Gb/s en distancias mayores de 1 metro mediante OFDM (Orthogonal Frequency Division Multiplexing) [3].

Utilizando distintos tipos de LEDs, diferentes tipos de fotoreceptores y modificando el tipo de modulación entre otras cosas, se han ido aumentando las tasas de transmisión alcanzándose en 2017 una tasa de 17.6 Gb/s a una distancia de 16 metros entre emisor y receptor. Esto se consiguió mediante un LD (Laser Diode) azul en el emisor, UPD (Ultrafast Photodiode) en el receptor y una modulación QAM (Quadrature Amplitud Modulation) con OFDM, en concreto el formato de QAM utilizado fue 16-QAM [4].

A continuación se muestra una tabla en la que se ven los avances en cuanto a tasa de transmisión y distancia de emisión en los últimos años. Aunque como ya se ha mencionado en la introducción, este no es el fin de este Trabajo Fin de Grado, ya que este es la construcción de un sistema de comunicaciones por luz visible a una velocidad baja comparada con los que se está llegando a conseguir en la actualidad. Respecto a esto hay varios documentos tales como Trabajos Fin de Grado realizados, en los que se busca un fin similar al nuestro.

Año	**Transmisor utilizado**	**Receptor utilizado**	**Tasa obtenida**	**Distancia enlace**	**Ref.**
2017	LD azul	UPD	17.6 Gb/s	16 m	[4]
2017	μ-LED	PIN	7.91 Gb/s	/	[5]
2016	RC-LED μ-LEDs	PIN	10Gb/s	15 m	[6]
2016	μ-LED	/	3.5 Gb/s	/	[7]
2015	RGB LDs	PIN	14 Gb/s	2.8 m	[8]
2015	Blue LDs	APD	9 Gb/s	5 m	[9]
2014	RGB LED	APD	4.22 Gb/s	/	[10]
2014	ps-LED	PIN	550 Mb/s	0.6 m	[11]

Tabla 2.1: Logros conseguidos en diferentes años en sistemas VLC

Observando la tabla 2.1 y como conclusión a esta sección, se puede ver como se han llegado a conseguir tasas de transmisión muy altas respecto a otras tecnologías, no siendo esta la única ventaja obtenible de este tipo de comunicaciones. Si realizamos una comparación rápida con las comunicaciones basadas en RF (Radio Frequency), las cuales son las más populares en los sistemas de comunicación modernos, las comunicaciones por luz visible tienen un ancho de banda aproximadamente 10000 veces más amplio [12], siendo esto una gran ventaja ante la saturación del espectro de RF en comunicaciones inalámbricas. Además el espectro de comunicaciones por luz visible no requiere autorización ni regulación a diferencia del de RF, pudiéndose transmitir sin ningún problema. Otras ventajas de este tipo de sistemas son la no producción de interferencias con otros dispositivos basados en RF, el bajo coste de los dispositivos con respecto a otros tipos de comunicaciones y una gran eficiencia.

2.3. Aplicaciones

Debido a las ventajas enumeradas en el final de la sección anterior, en algunos escenarios ya se ha optado por este tipo de comunicaciones, mientras que en otros ya se están empezando a introducir. A continuación se muestran algunas aplicaciones en las que se están llevando a cabo y algunos campos donde podrían utilizarse.

- **Li-Fi**: Se trata de un sistema VLC bidireccional análogo al Wi-Fi, en la actualidad es el sistema basado en este tipo de tecnología más conocido. A continuación se muestra una imagen de una situación de la vida cotidiana en la que nos serviría esta aplicación:

Figura 2.3: Ejemplo de un sistema Li-Fi

Mediante esta tecnología se pueden obtener velocidades por encima de 10 Gb/s, unas 250 veces más rápido que cualquier red de banda ancha ultra rápida [13]. Respecto al Wi-Fi, tecnología que todos utilizamos en nuestras casas para conectarnos a internet, una gran ventaja sería la ausencia de interferencias con otras tecnologías basadas en RF, aunque, como todos los sistemas, cuenta con desventajas, siendo la principal y de gran importancia la necesidad de visión directa entre emisor y receptor, como sucede en aquellos basados en VLC. Debido a esto, el Li-Fi podría considerarse como un complemento al Wi-Fi más que como un competidor, como podría parecer al visualizar la figura 2.3 si no conocemos cuáles son sus principales características.

- **Atención sanitaria**: Este sería otro campo en el que se está empezando a utilizar este tipo de comunicaciones, ya que la ausencia de interferencias con otras tecnologías como ya se ha ido mencionando a lo largo del texto, lo convierten en una tecnología de gran importancia en un campo tan complicado como la medicina. Un ejemplo de este tipo de comunicaciones dentro de este campo podría ser la monitorización de cualquier paciente a través de un sensor y el envío de esa información mediante VLC.

- **Comunicaciones submarinas**: Debido a que las señales de RF no pueden viajar con facilidad a través del mar, VLC podría ser una solución a algún caso dentro de las comunicaciones submarinas, como por ejemplo sería el envío de información entre submarinos, siendo la distancia un gran limitante para este tipo de aplicaciones, reduciéndolas a casos concretos como el de estos vehículos.

- **Transporte inteligente**: Otra de las tantas aplicaciones que soportaría este tipo de comunicaciones sería esta, con la que se podría enviar información entre

los diversos elementos de la vía tales como: conductores, semáforos o farolas. En la figura 2.4 se puede observar un ejemplo de este tipo de aplicación en la que nos da una idea de como sería el intercambio de información entre los diferentes elementos:

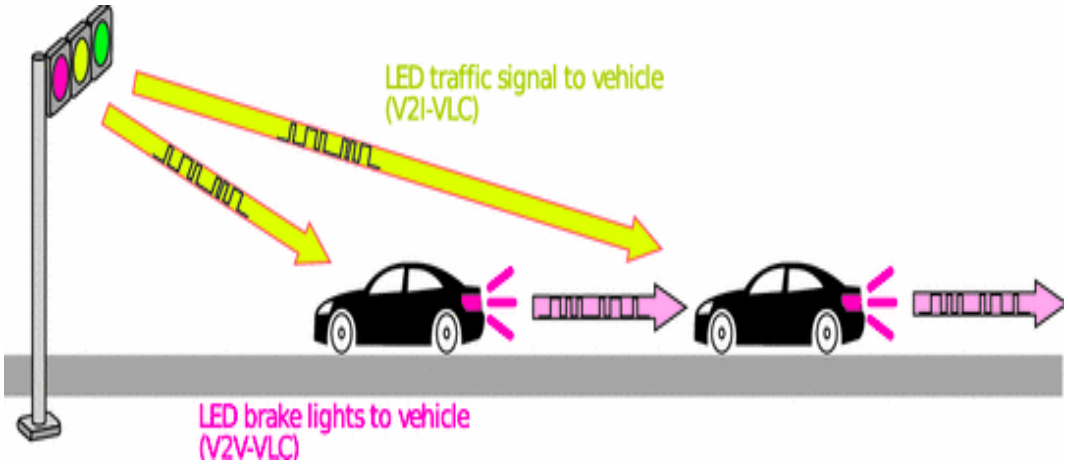

Figura 2.4: Aplicación VLC para el transporte inteligente

Esta información transmitida entre elementos viales podría ser de utilidad para, por ejemplo, la seguridad de los conductores. El mensaje recibido podría ser un frenazo de un vehículo que se encuentra por delante o la velocidad del que va detrás, permitiendo que el automóvil pudiera decidir qué hacer en cada situación. La información a transmitir sería infinita del mismo modo que las acciones a realizar con ella, abriendo un gran abanico de posibilidades.

- **Posicionamiento indoor**: Esta aplicación está siendo objeto de estudio dentro de las comunicaciones VLC dado que si usamos, por ejemplo, GPS (Global Positioning System), tendremos una pérdida de potencia en este debido a los obstáculos que ha de atravesar [14]. A continuación en la figura 2.5 se muestra el sistema basado en VLC utilizado en el artículo citado anteriormente, en el cual se obtiene la posición del objeto mediante el uso de 3 LEDs emitiendo y un detector.

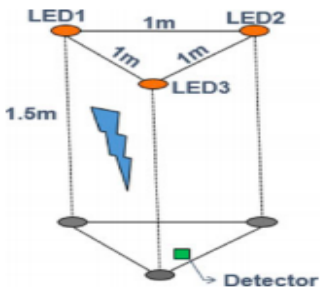

Figura 2.5: Sistema de posicionamiento indoor [14]

Capítulo 3

Diseño del sistema VLC

En este capítulo se va a desarrollar el sistema implementado tratando de diferenciar los distintos bloques que tendríamos si nos basásemos en un sistema de comunicaciones básico, pudiendo observar las diferentes soluciones tomadas durante el proyecto. Se explicará en primer lugar la parte analógica tanto de emisor como de receptor, para a continuación mostrar la parte digital implementada para los mismos.

3.1. Introducción

Para comenzar este capítulo, a continuación se muestra una figura representativa del sistema desarrollado, donde se pueden observar sus diferentes bloques:

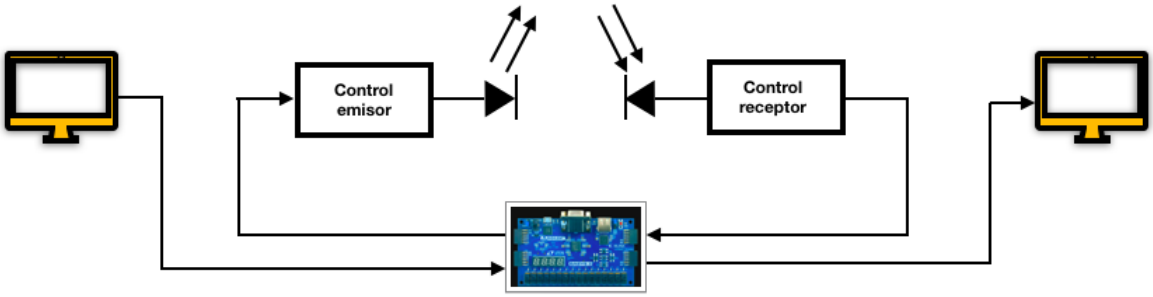

Figura 3.1: Diagrama de bloques del sistema desarrollado

Como podemos ver, tenemos un emisor del que forman parte un PC, una FPGA, el bloque llamado **Control emisor** y un LED. El canal, como en todos los sistemas basados en luz visible es el aire, mientras que desde el punto de vista del receptor tenemos un fotodiodo, el bloque llamado **Control receptor**, una FPGA y un PC. Destacar que tan solo se han usado una FPGA y un PC para recibir y transmitir.

A continuación se muestran dos figuras con las que se representa el sistema completo desarrollado en este Trabajo Fin de Grado, tanto desde el punto de vista analógico como digital. Los nombres y las señales que aparecen en los bloques anaranjados son los utilizados realmente para la implementación en VHDL, estos pueden ser consultados en linea como ya se mencionó en la introducción del Trabajo Fin de Grado. Conforme se avance en el capítulo, se irá haciendo referencia a estos y a sus señales con el fin de explicar más en detalle las funciones realizadas desde un punto de vista al más bajo nivel.

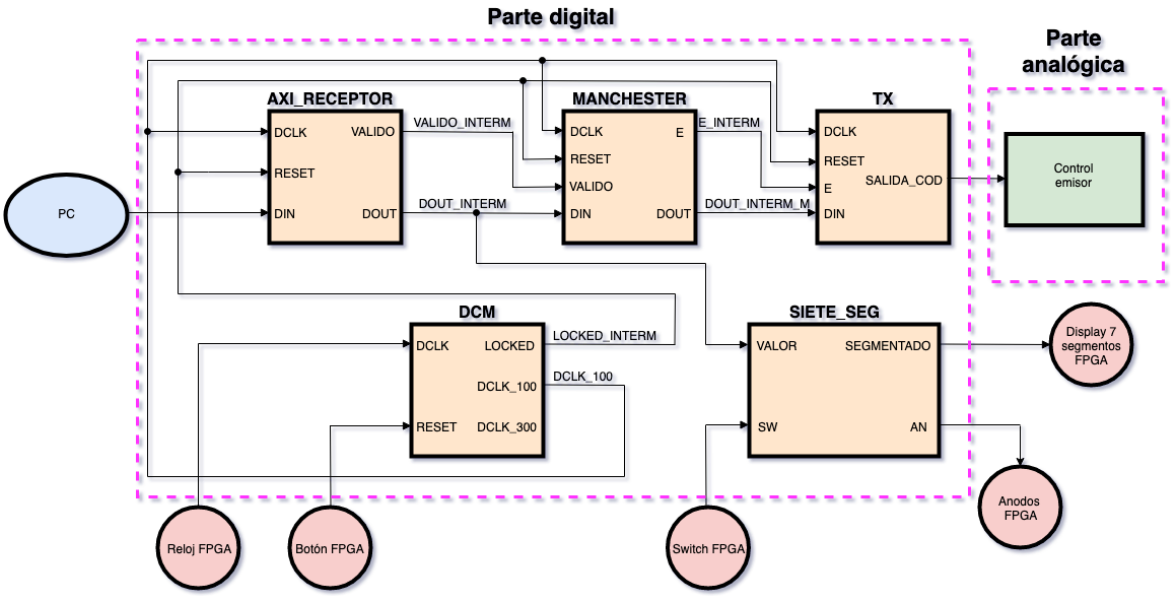

Figura 3.2: Diagrama de bloques correspondiente al emisor

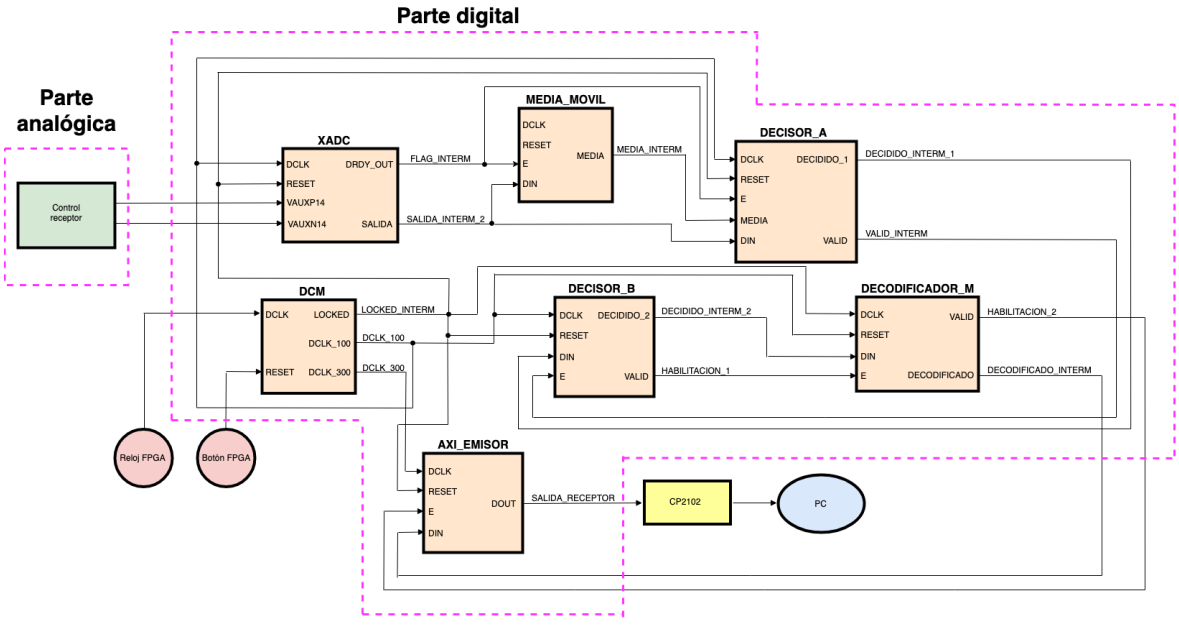

Figura 3.3: Diagrama de bloques correspondiente al receptor

3.2. Parte analógica

3.2.1. Emisor

Seguidamente se comienza la explicación del primer bloque analógico que se ha diseñado en este Trabajo Fin de Grado, el cual aparecen en la figura 3.2 con el nombre de **Control emisor** .

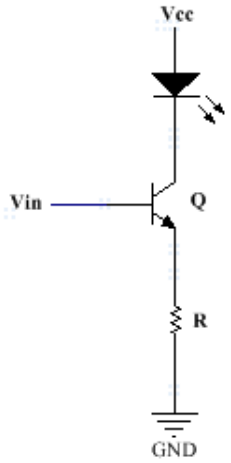

Figura 3.4: Circuito de control del emisor

En la figura 3.4 aparece el circuito diseñado con el objetivo de controlar la transmisión de los bits en el emisor. La alimentación del circuito, $+\mathbf{V_{cc}}$, se produce a 7 voltios, conectándose directamente al ánodo del diodo emisor. El diodo utilizado es de luz blanca de alta luminosidad, el fabricante de este es LUMEX. La hoja de características de los componentes utilizados se puede consultar en el correspondiente enlace dentro del **Anexo A**. Destacar además el **Anexo F**, donde se puede encontrar una pequeña explicación de las formas de generación de luz blanca, y el porqué del uso en este Trabajo Fin de Grado del tipo de LED seleccionado.

Continuando con el análisis del circuito, tenemos un transistor, en cuya base se introduce la señal proveniente de la FPGA, $\mathbf{V_{in}}$. Esta será una serie alterna de '0' y '1' que provocará que el transistor esté continuamente conmutando entre la zona activa y de corte. Otra función proporcionada será el suministro de la corriente necesaria para que el LED luzca de forma correcta. La resistencia, **R**, de valor 390 Ohmios, conectada a la pata emisora, se utiliza para que toda la corriente que fluye por el circuito no vuelva hacia la placa y nos la dañe.

3.2.2. Receptor

Mediante este bloque se lleva a cabo la recepción de la señal proveniente del emisor con el propósito de acondicionarla y poder procesarla dentro de la FPGA. Este, dentro del esquema de la figura 3.3, aparece con el nombre **Control receptor**. A continuación se realiza una representación de los componentes que tiene este bloque, explicándose más adelante la finalidad de cada uno.

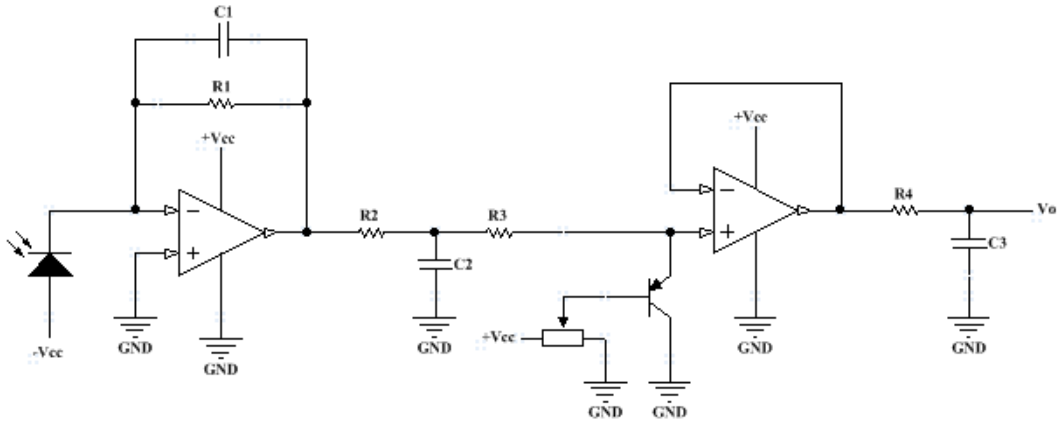

Figura 3.5: Circuito de recepción analógico

Partiendo de la necesidad de convertir la señal óptica que viaja por el aire en una eléctrica con la que podamos trabajar, se hace uso de un fotodiodo operando en modo fotoconductivo (polarización inversa), obteniendo así una corriente proporcional a la luz incidente. En la figura 3.5 se puede observar cómo el ánodo del fotodiodo se conecta a $-\mathbf{V_{cc}}$, -7 voltios, asegurándonos así tener el fotodiodo trabajando en el modo mencionado. El componente utilizado ha sido el **VTB-1013BH** del que se va a destacar su sensibilidad. En el **Anexo G** se pueden ver los diferentes modos de operación del fotodiodo junto con los tipos, justificando el porqué de nuestra elección. Continuando con la sensibilidad del fotodiodo, se muestra a continuación una figura de la misma, donde podemos observar cómo la respuesta del fotodiodo seleccionado abarca completamente el espectro de la luz visible.

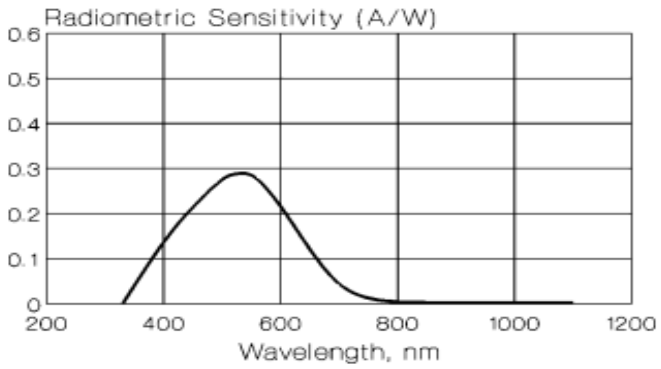

Figura 3.6: Representación de la sensibilidad del fotodiodo

El enlace a la hoja de características del fotodiodo se puede consultar en el **Anexo A** en la parte correspondiente a este, lugar de referencia de donde se ha tomado la figura anterior.

Una vez realizada la conversión óptico-eléctrica, el siguiente paso a llevar a cabo es la amplificación de la señal. Para ello se ha hecho uso de un amplificador de transimpedancia que actúa como conversor corriente-voltaje. Con este fin se ha usado el **LM-358**, un amplificador operacional doble de propósito general, en el que, como podemos ver en la figura 3.5, se conecta el cátodo del fotodiodo a la entrada inversora del operacional. Para realizar la amplificación se coloca a modo de realimentación una resistencia, en este caso de 470 kΩ, añadiéndose además un condensador de 33 pF en paralelo con el fin de estabilizar la señal, llevando a cabo por tanto la amplificación y un filtrado de tipo paso-bajo.

Decir que para esta primera parte del esquema de la figura 3.5 nos hemos basado en otros trabajos [15], [16], acondicionándolos a las características de nuestra señal. Además destacar también las diferentes alternativas a la hora de seleccionar el fotorreceptor, siendo el fotodiodo el más adecuado para el caso de VLC debido a su mayor velocidad de respuesta frente a otros fotorreceptores como el fototransistor o la fotorresistencia. Los diferentes fotorreceptores junto con los componentes necesarios para su acondicionamiento, además de la pertinente explicación, se pueden ver en la tesis correspondiente a la referencia [15].

Continuando con el esquema de la figura 3.5 se ha diseñado un filtro antialiasing para el ADC (Analogic to Digital Converter) de la FPGA. Para ello se ha hecho uso del esquema de la figura mostrada a continuación.

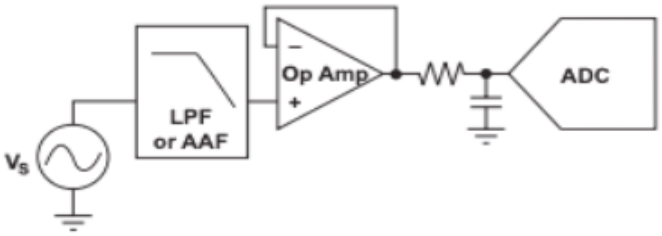

Figura 3.7: Esquema general de un sistema de adquisición de datos [17]

En el esquema anterior podemos observar cómo la señal que tenemos debe ser en primer lugar conectada a un AAF (Antialiasing Filter). En nuestro caso esto se ha realizado conectando a un filtro paso bajo de primer orden la señal correspondiente a la salida del amplificador operacional. En la figura 3.5 el filtro estaría formado por los componentes $\mathbf{R_2}$ y $\mathbf{C_2}$, cuyos valores son de 1.2 MΩ y 12 pF respectivamente. Con este filtro obtenemos un frecuencia un poco por encima de los 11 kHz, $\frac{1}{2*\pi*R*C}$, asegurándonos así la ausencia de aliasing, ya que la frecuencia de muestro del conversor es de 77 kHz. Mencionar la existencia de la herramienta **Filter Design Tool** de Texas Instruments con la que se puede diseñar cualquier tipo de filtro, seleccionando parámetros como la frecuencia deseada, rizado, ganancia, orden del filtro, atenuación, etc. Aunque se ha valorado esta opción, se ha optado por algo sencillo y rápido de implementar.

Respecto del esquema de la figura 3.7, en nuestro circuito se ha añadido a continuación del AAF un circuito para controlar la tensión de entrada a la FPGA, puesto que el conversor de la misma digitaliza valores de tensión entre 0 y 1 voltio, pudiendo causar daño en ella si introducimos voltajes por encima de este. A continuación se muestra el esquema del circuito de control de voltaje, el cual se ha extraído de la figura 3.5.

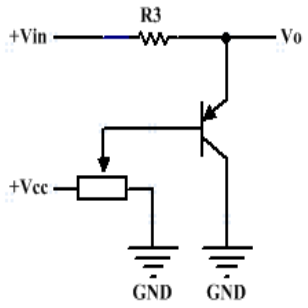

Figura 3.8: Circuito controlador de voltaje

El funcionamiento del circuito de control se realiza fijando una tensión en la base del transistor a través de un potenciómetro. Esta tensión deberá ser tal, que en caso de que tengamos en el emisor del transistor una tensión mayor a 1 voltio, este conmute pasando de la zona de corte a la zona activa, consiguiendo así que a la salida del circuito, $\mathbf{V_0}$, tengamos un valor de 1 voltio, máximo digitalizable por la FPGA. En el caso contrario, en el cual la señal en el emisor no supera 1 voltio, el transistor permanecerá en corte, teniendo así en $\mathbf{V_0}$ la misma señal que a la entrada $\mathbf{V_{in}}$ pero habiendo atravesado una resistencia, $\mathbf{R_3}$. Respecto a los componentes utilizados, el modelo del transistor es el **BC557**, un PNP de propósito general, el potenciómetro se trata del **T93VB** y el valor de la resistencia utilizada, $\mathbf{R_3}$, es de 680 Ω.

Continuando con el esquema, una vez introducido el circuito de control de voltaje, seguimos con el diseño del DAQ (Data Acquisition) representado en la figura 3.7. Lo siguiente que aparece en esta es un seguidor de tensión, con este para la FPGA todo el circuito montado a su izquierda será invisible, ya que verá una impedancia muy grande, teóricamente infinita. Así conseguimos no tener problemas de desadaptación de impedancias, con las pérdidas de potencia en la señal correspondientes. Las alimentaciones de todos los operacionales del circuito de la figura 3.5 se realizan a $\pm\mathbf{V_{cc}}$.

Por último, en la figura 3.5, finalmente se monta un filtro paso bajo, de primer orden, mediante la resistencia $\mathbf{R_4}$ y el condensador $\mathbf{C_3}$, siendo los valores de 220 kΩ y 1,5 pF respectivamente. La salida de este filtro paso bajo será conectada directamente al ADC de la FPGA, cuyo circuito de entrada se representa a continuación.

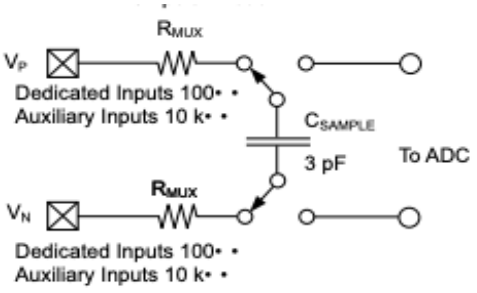

Figura 3.9: Circuito de entrada analógico equivalente para las entradas auxiliares [18]

En nuestro caso, al usar entradas auxiliares, la resistencia de entrada al conversor, $\mathbf{R_{MUX}}$, será de 10 kΩ, mientras que el condensador que se encarga de realizar el muestreo, $\mathbf{C_{SAMPLE}}$, es de 3 pF para todos los casos. Las hojas de características de cada componente, igual que antes, se podrán encontrar en el enlace correspondiente

dentro del **Anexo A**.

En la sección siguiente se profundizará entre otras partes en el ADC, explicando el porqué del uso de las entradas auxiliares y no de las dedicadas.

3.3. Parte digital

3.3.1. Emisor

A continuación se muestra una figura representativa de los bloques que incluye esta parte del proyecto. A lo largo de esta sección se tratará de explicar cada uno en profundidad intentando aclarar su función dentro de este sistema. Remarcar, por otro lado, la importancia de haber utilizado para realizar la transmisión de los caracteres la modulación OOK con codificación tipo Manchester, ya que, como se explicará en los bloques correspondientes, nos aportará a nuestra implementación una gran cantidad de ventajas.

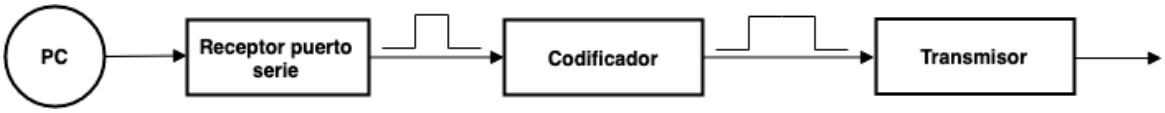

Figura 3.10: Diagrama de bloques digitales correspondientes al emisor

Receptor puerto serie

Como se puede observar en la figura 3.2, dentro del diagrama general del emisor, este bloque se corresponde con **AXI_RECEPTOR**. En la figura 3.10 se muestra cómo los datos transmitidos desde el PC son mandados por el puerto serie, necesitando así un bloque que se encargue de su recepción. De esto nos ocuparemos en esta parte del proyecto. Para ello, se hará uso del IP conocido como **AXI Uartlite** con el que realizaremos la recepción puerto serie. Con este se puede llevar a cabo tanto la recepción puerto serie como la transmisión, lo cual dependerá de las señales internas que se utilicen. Para su uso deberemos realizar su instanciación. A continuación se mostrará un ejemplo de cómo realizar este proceso, siendo el mismo tanto si queremos hacer uso de la recepción como de la transmisión puerto serie.

Para llevar a cabo la instanciación dentro de nuestro proyecto, deberemos ir al apartado **Project manager>IP Catalog**, lo cual nos abrirá un listado de IP Cores disponibles para utilizar. Una vez aquí la ruta hasta el **AXI Uartlite** es la siguiente **Vivado Repository>Embedded Processing>AXI Peripheral>Low Speed Peripheral**. Después de seleccionar esta pestaña nos aparecerá un asistente

que nos permitirá la elección de algunas opciones. A continuación se muestra una imagen de lo mencionado:

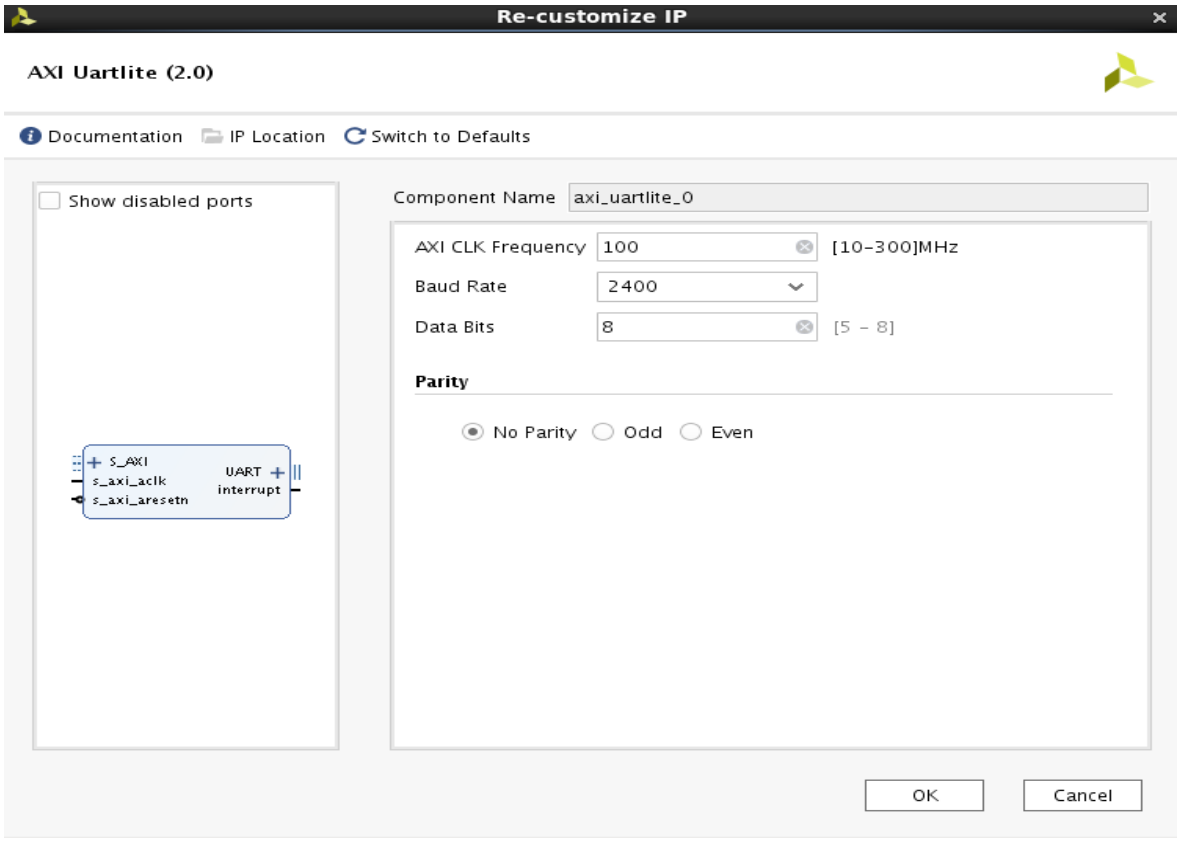

Figura 3.11: Asistente de configuración de la Uartlite

En la figura 3.11 se observa la configuración seleccionada en este caso para la recepción puerto serie, siendo la tasa de recepción la misma a la que se producirá la transmisión de los bits en nuestro sistema, 2400 b/s. Como ya se ha nombrado anteriormente, la configuración del IP se realiza a través de sus señales internas. En la figura 3.12 se pueden observar las que intervienen en el proceso de configuración de la recepción puerto serie, viéndose los valores que debemos darle a lo largo del proceso.

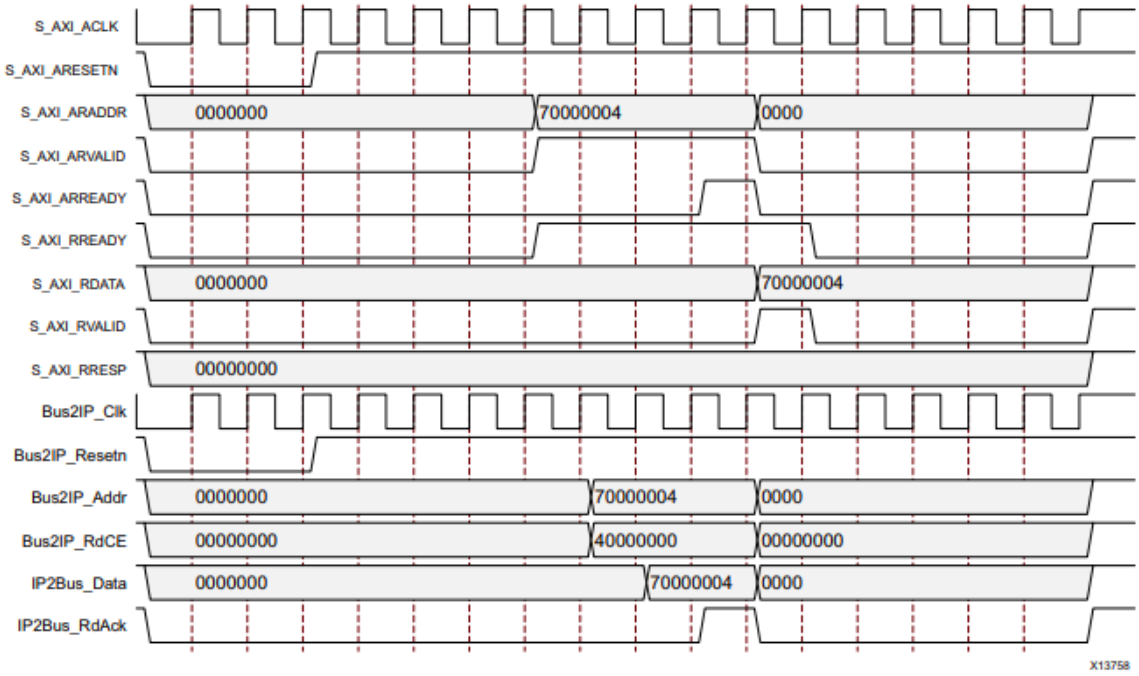

Figura 3.12: Operación de lectura del AXI [19]

A continuación se realiza una explicación más detallada del proceso general para más adelante explicar el uso concreto que le hemos dado a este IP dentro de nuestro Trabajo Fin de Grado y los registros que hemos utilizado.

1. El maestro escribe una dirección en la señal **S_AXI_ARADDR** para a continuación marcarla como válida mediante la validación de **S_AXI_ARVALID**. En este momento estaremos en disposición de indicarle al esclavo que estamos preparados para la recepción de datos, para ello deberemos validar **S_AXI_RREADY**.

2. El esclavo afirmará **S_AXI_ARREADY**, indicando que está preparado para la recepción de la dirección en el bus, siendo necesario mantener los valores anteriores hasta que el esclavo confirme la señal. En el momento en el que tanto **S_AXI_ARVALID** como **S_AXI_ARREADY** estén validadas se lleva a cabo la operación, en la que el esclavo se guarda la dirección que ha escrito el maestro. A continuación estas señales son puestas a '0' respectivamente.

3. El esclavo coloca los datos en el canal y pone a '1' **S_AXI_RVALID** indicando que el dato en el canal es válido.

4. Cuando **S_AXI_RREADY** y **S_AXI_RVALID** están a '1', se lleva a cabo la operación, pudiendo ser en ese momento puestas de nuevo a '0', su estado inicial.

El proceso descrito se puede encontrar en la página correspondiente a la referencia [20]. Respecto a nuestro caso, el proceso realizado es similar al descrito, escribiendo las direcciones de los diferentes registros a los que queremos acceder en la señal **S_AXI_ARADDR**. En la figura 3.13 se pueden observar las diferentes direcciones de los registros accesibles.

Address Offset	Register Name	Description
0h	Rx FIFO	Receive data FIFO
04h	Tx FIFO	Transmit data FIFO
08h	STAT_REG	UART Lite status register
0Ch	CTRL_REG	UART Lite control register

Figura 3.13: Registros y direcciones del AXI UARTLITE [21]

Para la lectura del puerto serie en este Trabajo Fin de Grado se ha hecho uso de los registros de estado **STAT_REG**, el cual nos da información sobre el estado de la cola de datos, y de **Rx FIFO**, donde tenemos los datos correspondientes a la recepción puerto serie. Respecto del registro de estado, hemos utilizado la posición 0 del vector con la finalidad de conocer si tenemos un dato recibido válido en cola. Tendremos un dato válido cuando en la posición mencionada haya un '1'. Mientras que el valor no sea este no deberemos comenzar el proceso de lectura del puerto serie. A continuación se muestra una figura donde podemos ver lo descrito anteriormente, así como las diferentes opciones que nos ofrece este registro.

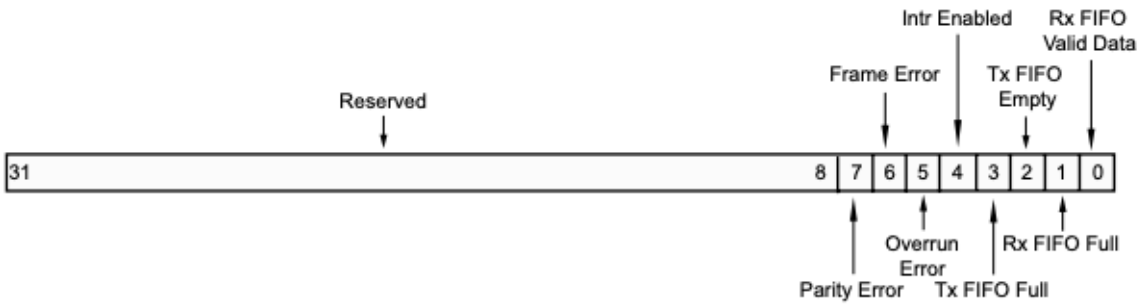

Figura 3.14: Registro de estado del AXI UARTLITE [21]

Como ya se ha mencionado y podemos observar en la figura 3.14, se ha utilizado la posición del vector correspondiente a **RX FIFO Valid Data**, para conseguir una lectura de datos correcta. De las demás opciones que nos da el registro no se ha hecho uso, ya que no se han considerado necesarias para lo que queríamos hacer. Para acceder a este registro la dirección que deberemos escribir en la señal

S_AXI_ARADDR será "08" en hexadecimal, como se puede observar en la figura 3.13

El otro registro utilizado es **Rx FIFO** y, de la misma manera que sucede con el anterior, se adjunta una figura para una mejor explicación.

Figura 3.15: Registro de lectura del AXI UARTLITE [21]

En la figura 3.15 podemos ver cómo los datos recibidos, **Rx Data**, se encuentran en los 8 bits menos significativos de este registro. Para realizar el acceso a este deberemos escribir en hexadecimal la dirección "00" en la señal **S_AXI_ARADDR**. Igual que antes, esta dirección se obtiene de la tabla correspondiente a la figura 3.13.

Para concluir, cabe destacar que el proceso referente a la figura 3.12 se deberá hacer para las direcciones de ambos registros mencionados, finalizando todo el proceso de recepción puerto serie una vez leídos los 8 bits menos significativos del registro **Rx Data**, como hemos podido ver en la figura 3.15. Con el fin de que el lector obtenga una idea más clara del proceso descrito anteriormente, se muestra a continuación una figura en la que aparece la máquina de estados diseñada para controlar el IP.

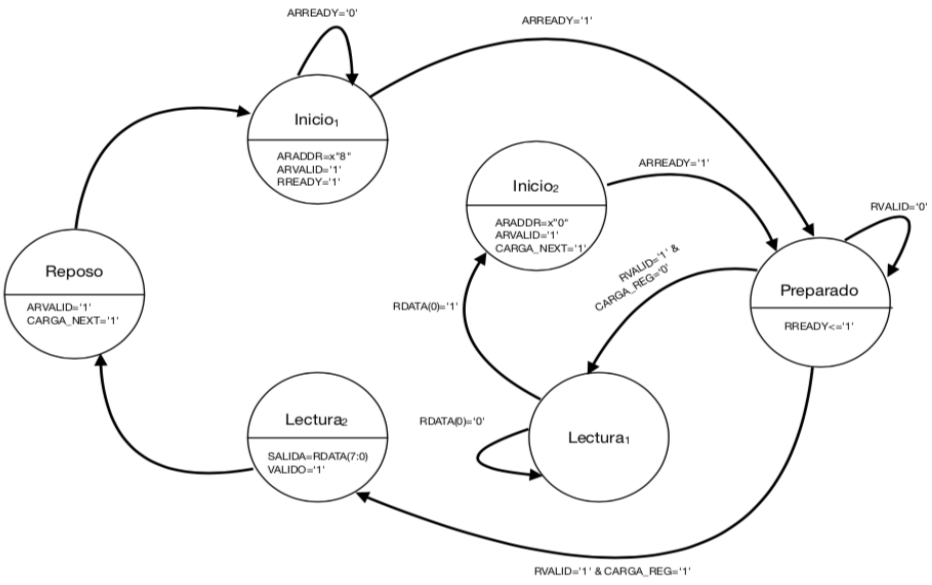

Figura 3.16: Máquina de estados encargada de controlar el IP usado en la recepción puerto serie

Codificador

Como se puede observar en la figura 3.2, dentro del diagrama general del emisor este bloque es el encargado de realizar la codificación tipo Manchester de los bits recibidos por el puerto serie. Esto se producirá cuando este indique con la señal **VALID_INTERM** (figura 3.2) la existencia de un nuevo dato leído del puerto serie. En la figura 3.17 se puede observar la codificación realizada para una secuencia ejemplo.

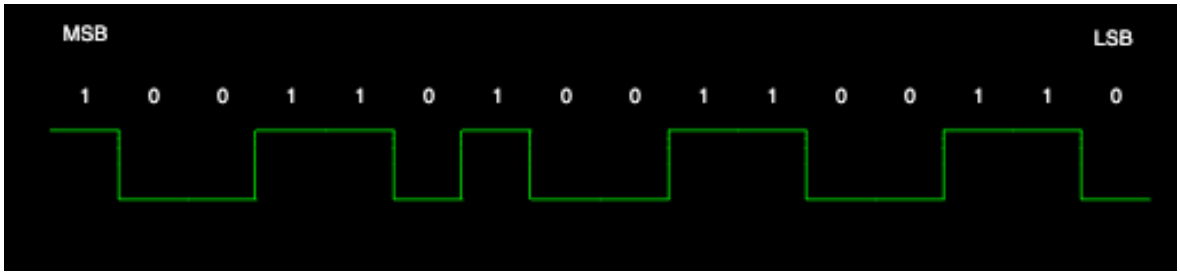

Figura 3.17: Codificación Manchester realizada para la secuencia de bits “100101”

Destacar respecto de la figura que el bit más significativo MSB (More Significative Bit) es el de más a la izquierda, siendo el de más a la derecha por tanto el menos significativo LSB (Less Significative Bit). La codificación se ha realizado partiendo del bit menos significativo LSB al más significativo MSB: el bit ’0’ será, por tanto, “01” y el ’1’ será “10”. Al realizar la codificación de este modo, pasaremos de una secuencia de 8 bits a otra de 16.

Una vez se haya concluido el proceso de codificación y tengamos los 16 bits válidos, se escribirán en la señal **DOUT_INTERM** para ser mandados al siguiente bloque, a la vez que se indica la validez del dato con la señal **E_INTERM**, poniendo esta a ’1’. Estas señales, igual que antes, se pueden observar en la figura 3.2.

Hacer un inciso para destacar este tipo de codificación dentro de los sistemas VLC como se mencionó al inicio de la sección. Con esta evitamos un gran problema que podría aparecer en este tipo de comunicaciones, el *flicker*, parpadeo producido por largas series de ’1’ o ’0’. En contraposición, aumentamos el ancho de banda al doble.

Por último, se muestra la máquina de estados diseñada para el control de este bloque, en la que el valor de **DOUT**, en el estado **Codif**, dependerá de si tenemos un ’1’ o un ’0’. Remarcar que esta es una aproximación de la desarrollada en VHDL, por lo que las señales no son las mismas que en el fichero .vhd correspondiente.

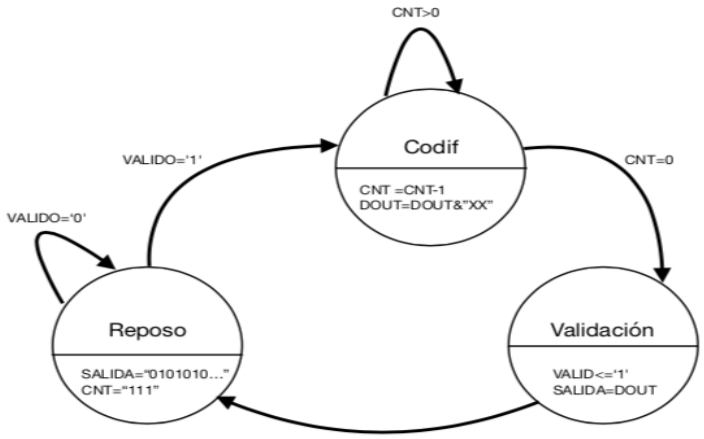

Figura 3.18: Máquina de estados encargada de realizar la codificación Manchester en nuestro sistema

Transmisor

Otro bloque que tenemos dentro del emisor es este, correspondiente a **TX** en la figura 3.2, donde se realiza la transmisión de los 16 bits mandados por el bloque anterior, **Codificador**. Esto se efectúa cuando la señal **E_INTERM** toma valor '1'. La transmisión se lleva a cabo a 2400 b/s, teniendo que mandar un bit cada 416 us, $\frac{1}{2400}$ segundos, lo que implica tener un contador de $\frac{100*10^6}{2400}$ muestras.

A continuación, aparece una imagen en la que podemos ver la línea de reposo del sistema, que será una secuencia de '1' codificados en Manchester:

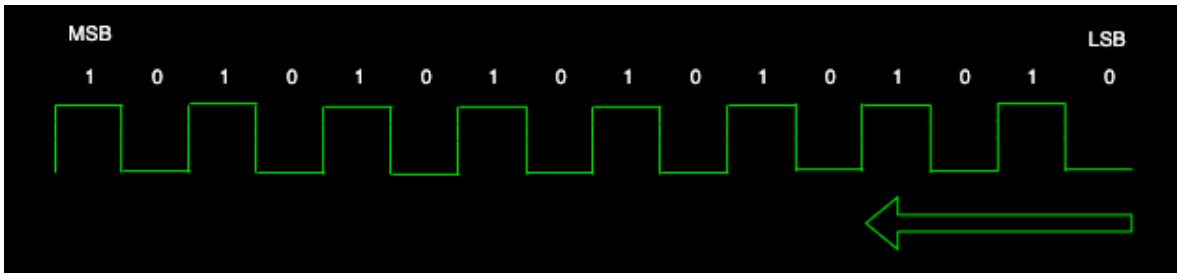

Figura 3.19: Línea de reposo codificada

Podemos observar en la figura como el bit '1' está codificado como "10". Además, se muestra una flecha que indica la dirección en la que son transmitidos los bits, siguiendo las mismas normas que en el bloque anterior. Estos bits serán los diferentes valores que tomará la señal **SALIDA_COD** mientras el sistema esté en reposo. Una vez que tengamos la señal **E** puesta a '1', indicándonos que tenemos una nueva secuencia de 16 bits a transmitir, le añadimos a esta un bit de START y otro de STOP codificados en Manchester con el fin de que en el receptor sepamos el comienzo y el fin

de una nueva secuencia de bits.

A continuación, se ha creado una figura para representar el proceso descrito. La secuencia graficada será la misma que la de la figura 3.17.

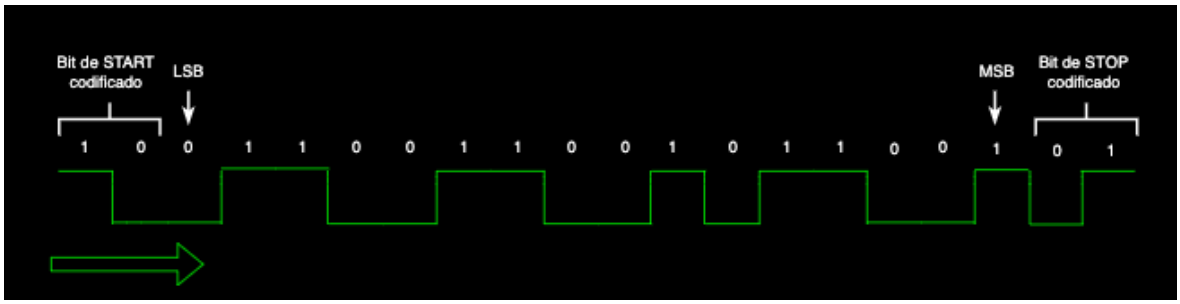

Figura 3.20: Formato de transmisión

Con este formato utilizado, rompemos la alternancia de '0' y '1' propia de la línea de reposo, ya que en el momento en el cual tengamos un nuevo carácter, la introducción del bit de START nos permitirá tener la secuencia"0110". Esos dos '1' le darán la capacidad al receptor de sincronizarse y comenzar el proceso de decodificación.

Como se ha mencionado anteriormente, la transmisión se realiza bit a bit mediante la señal **SALIDA_COD**, la cual será mapeada a los módulos periféricos de la placa Pmod (Peripherical module). Estos serán conectados al bloque que aparece en la figura 3.2 como **control emisor** con la simple ayuda de un cable, conectándose este a la entrada $\mathbf{V_{in}}$ mostrada en la figura 3.4.

Es importante destacar que cuando se mande un carácter por **SALIDA_REG** se negará cada bit mediante una puerta **not**. Esto se debe a que el circuito analógico correspondiente al receptor invierte la señal transmitida. De este modo conseguiremos que el receptor reciba un "11" para sincronizarse como ya se ha mencionado en el párrafo anterior.

Para concluir, igual que en los bloques anteriores, se muestra un diagrama de la máquina de estados diseñada para realizar la tarea de transmisión, en el que se puede observar el contador mencionado al principio de la descripción de este bloque, cuyo número de muestras es de 41666, $\frac{100*10^6}{2400}$.

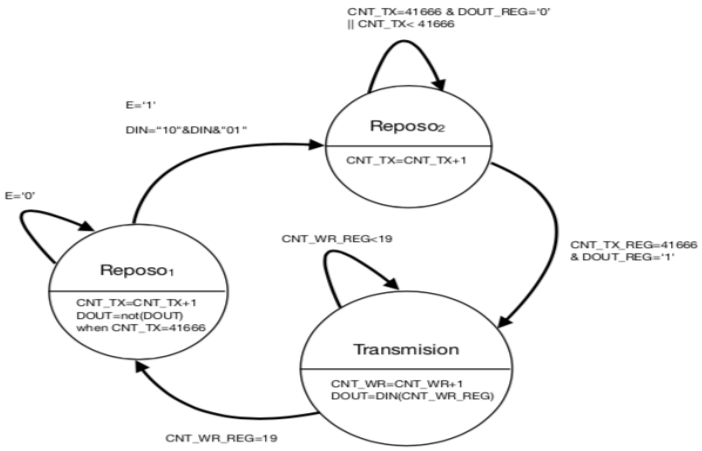

Figura 3.21: Máquina de estados encargada de realizar la transmisión a 2400 b/s en nuestro sistema

Visualizador 7 segmentos

Este bloque se utiliza como complemento a los otros dentro del emisor, es decir, no es necesario en el funcionamiento del enlace montado en este Trabajo Fin de Grado, por ello se ha decidido no incluirlo en la figura 3.10. La función de este será revelar los números recibidos por el puerto serie, pudiendo seleccionar además mediante los switches de la placa el ánodo por el cual queremos mostrar el número. Para esto se han usado los switches 0, 1, 2 y 3 de la placa. Las correspondientes conexiones de este bloque se pueden observar en la figura 3.2 con el nombre de **SIETE_SEG**.

3.3.2. Receptor

De la misma forma que para el emisor, en el receptor se comienza con una figura en la cual se representan los bloques contenidos en este. Para la explicación se seguirá la misma metodología utilizada hasta ahora, realizando una caracterización de cada bloque en profundidad. Será en esta etapa donde se puedan observar todas las ventajas aportadas por la modulación OOK basada en codificación Manchester, aparte de las ya mencionadas para los sistemas VLC.

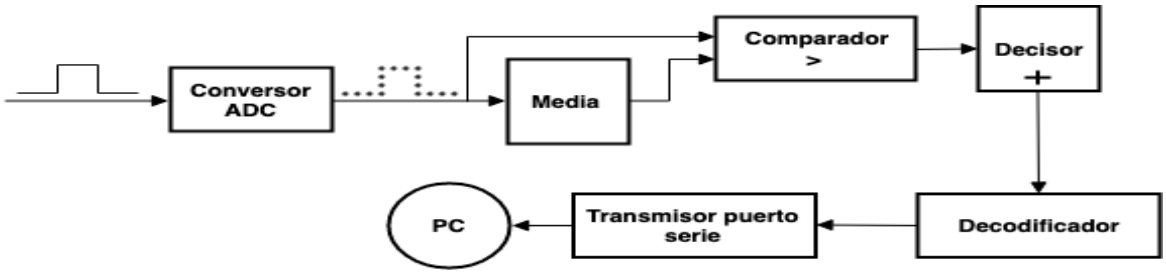

Figura 3.22: Diagrama de bloques digitales correspondientes al receptor

ADC

En el momento en el que la señal es acondicionada a nuestro sistema digital, salida $\mathbf{V_0}$ correspondiente a la figura 3.5, con el fin de procesar los datos recibidos mediante la FPGA, nos vemos obligados a llevar a cabo una conversión A/D. Para esta tarea usaremos el XADC, un IP mediante el cual podremos realizarla sin necesidad de conectar ningún dispositivo externo a nuestra placa. Para el uso de este IP se tienen varias opciones, en nuestro caso se ha decidido usarlo mediante el instanciamiento del mismo en nuestro diseño. Este proceso se hará igual que en los IP anteriores, solo que en este caso el nombre que tendremos que buscar es **XADC**. La ventana del asistente para este IP se muestra a continuación.

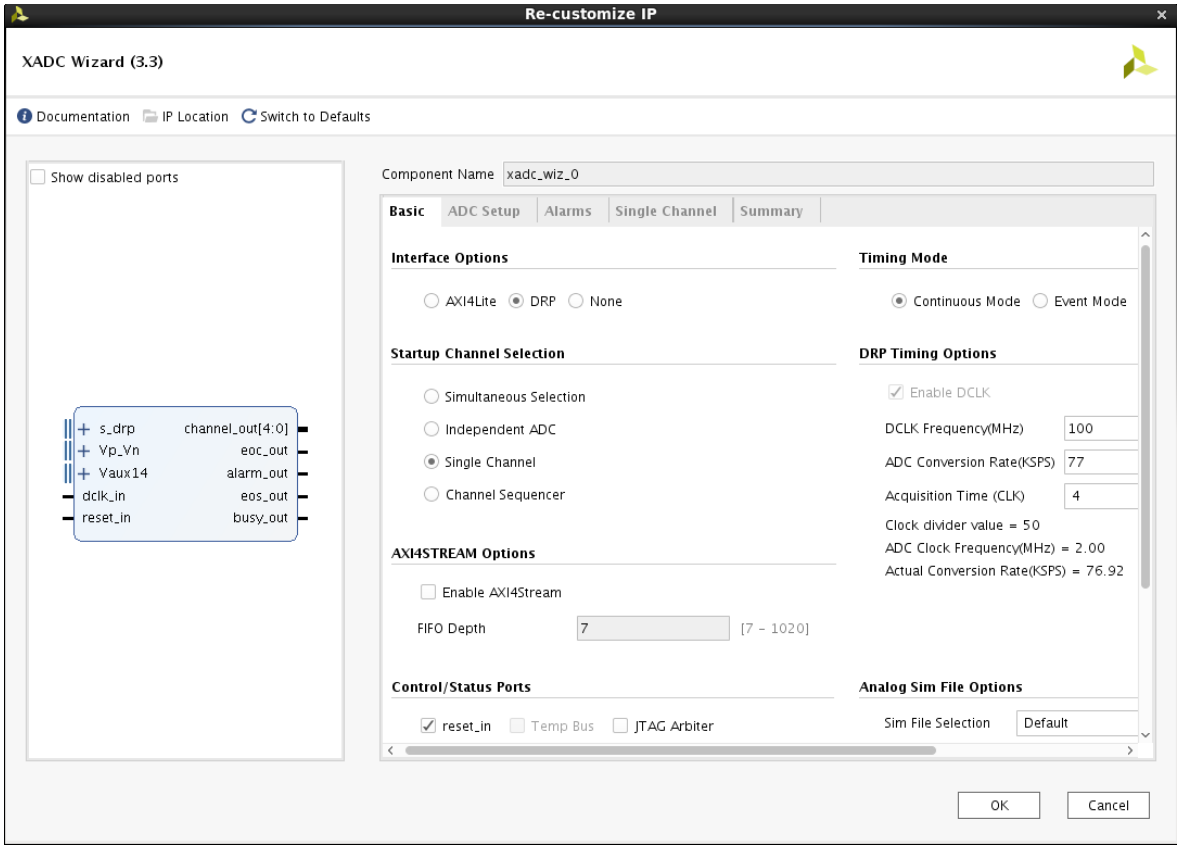

Figura 3.23: Asistente de configuración del XADC

Debido a la extensión de esta sección se va a optar por dividirla en diferentes subsecciones con el fin de que el lector obtenga una mayor claridad en la explicación. Se comenzará a continuación con las opciones seleccionadas en el IP, posteriormente se explicarán las señales que hemos usado para su funcionamiento para finalmente dar una explicación de este teniendo el lector una idea previa de lo realizado. Situar este bloque dentro de la figura 3.3 con el nombre de **XADC**.

Opciones seleccionadas para la configuración del XADC

En esta parte del trabajo se detallan las opciones de configuración del XADC seleccionadas en nuestro proyecto. Esto se ha realizado a través del asistente de configuración del IP, el cual se puede observar en la figura 3.23. Dentro de este aparecen una serie de pestañas que serán enumeradas a continuación y explicadas en detalle más adelante, ya que las opciones seleccionadas en este bloque serán de gran relevancia para el resto del proyecto. Las opciones que nos mostraría el asistente de configuración son:

- **Basic**
- **ADC Setup**
- **Alarms**
- **Single Channel**
- **Summary**

Comenzando con la pestaña **Basic**, en la opción **Interface options**, se ha seleccionado **DRP**, cuyas señales son utilizadas con el fin de controlar la conversión del XADC, más adelante se profundizará en mayor medida en este bloque. La otra opción que nos aparece dentro de **Interface options** es **AXI4Lite**, esta se ha utilizado dentro de nuestro Trabajo Fin de Grado en los IP correspondientes al receptor puerto serie, **AXI_RECEPTOR**, y en el transmisor puerto serie, el cual será explicado en secciones próximas. Aquí se planteó su uso pero finalmente debido a su sencillez se optó por la interfaz **DRP**.

Continuando en la pestaña **Basic**, destacar la selección de **Single Channel** dentro de **Startup Channel Selection**. Con esto se consigue realizar la conversión de la señal de un único canal, en nuestro caso la introducida por el canal auxiliar número 14. Dentro del apartado **Control/Status Ports** se ha habilitado **reset_in** con el que tenemos una señal de reset que usaremos para conseguir sincronismo en el sistema. Dentro de la opción **Timming Mode** se ha seleccionado **Continuous Mode**, consiguiendo así tener el XADC trabajando en modo continuo, es decir, realizando la conversión A/D de la señal en todo momento, lo cual nos evita complicaciones respecto al inicio de la conversión de la señal en nuestro sistema.

Por último nos queda ajustar los valores de la opción **DRP Timming Options** en la que dentro de **DCLK Frequency (MHz)** debemos introducir la frecuencia en MHz que va a tener el reloj de entrada al XADC, en nuestro caso

100 MHz. A continuación nos aparece **ADC conversion Rate (KSPS)**, donde deberemos introducir la frecuencia de conversión del sistema, la cual será en nuestro caso aproximadamente de 77 KSPS, lo que significa que llevándose a cabo la trasmisión desde el emisor a una tasa de 2400 b/s, cada vez que se realice el muestreo obtendremos aproximadamente 32 muestras para determinar el bit recibido en los siguientes bloques, $\frac{77*10^3}{2400}$.

La siguiente y última pestaña seleccionada es **Single Channel**, donde debemos indicar el canal en el cual vamos a introducir la señal de la que queremos realizar la conversión A/D. Del resto no se ha considerado su utilización en este Trabajo Fin de Grado. Destacar finalmente la pestaña **Summary**, en la que podemos ver las opciones más relevantes seleccionadas en las anteriores.

Señales internas usadas para el control del IP

Continuando con la subdivisión del IP, se observa a continuación una figura en la que aparecen todas las señales internas del mismo. En este subapartado se tratarán de explicar una por una las que se han utilizado en nuestro Trabajo Fin de Grado, para que más adelante, cuando se realice la explicación del funcionamiento del IP, se conozca la función de cada señal.

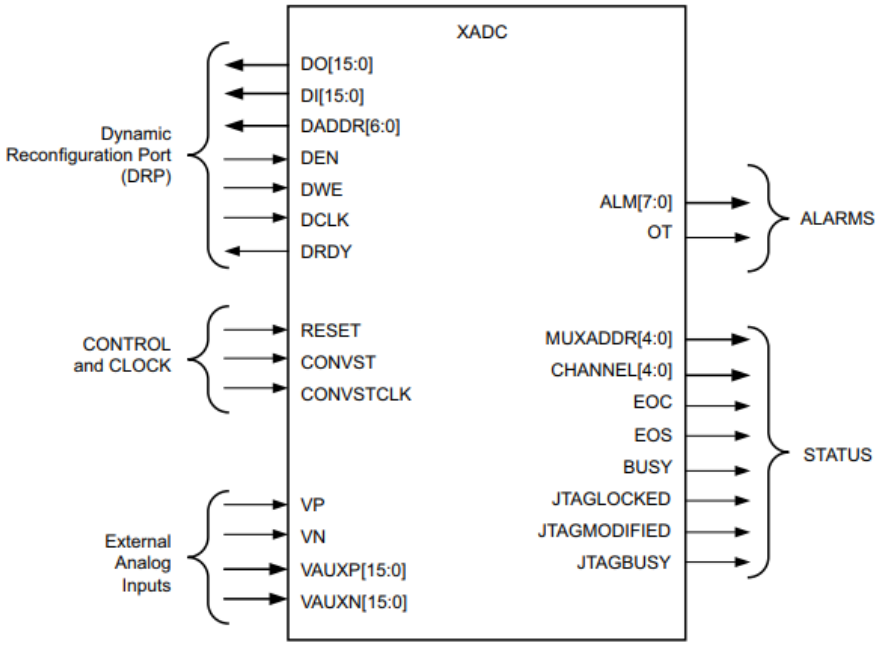

Figura 3.24: Puertos XADC [18]

Como ya se mencionó, para la realización del control del XADC se han usado las señales del DRP (Dynamic Reconfiguration Port), aunque no todas ellas nos son de utilidad. A continuación se muestra una breve descripción de los puertos usados en nuestro proyecto, tanto los del DRP como los propios del XADC:

- **DCLK**: Reloj que deberemos introducir al DRP. Hará la función de reloj del XADC con una frecuencia de 100 MHz. Esta señal es generada a través del IP encargado de crear los relojes del sistema.
- **RESET**: Reset del XADC, generado externamente por el mismo IP con el que se crean los relojes del sistema.
- **VAUXP[15:0]**: Vector de 16 señales analógicas positivas, de las cuales nuestra placa, Basys 3, solo tiene habilitadas la **VAUXP6**, **VAUXP7**, **VAUXP14** y **VAUXP15**. En nuestro caso se ha usado la señal **VAUXP14**, por la que introduciremos la señal de la cual deseamos realizar la conversión A/D.
- **VAUXN[15:0]**: Vector de 16 señales analógicas negativas. La casuística es la misma que para las señales positivas. Será conectada a tierra.
- **DRDY**: Puerto perteneciente al DRP que nos indicará la disponibilidad de los datos convertidos. Tomará valor '1' cuando los datos sean estables; de esta forma se notificará a los siguientes bloques la tenencia de un nuevo dato válido.
- **DEN**: Puerto perteneciente al DRP con el cual indicaremos al XADC que queremos hacer un acceso de lectura.
- **EOC**: Puerto del XADC que indica cuándo ha finalizado la conversión y ha sido escrito el valor de esta en el registro de estado.
- **DO[15:0]**: Puerto perteneciente al DRP donde el XADC pone en los 12 bits más significativos los datos correspondientes a la conversión A/D.
- **DADDR[6:0]**: Puerto de 7 bits perteneciente al DRP en el que indicamos la dirección del registro de estado que queremos leer. Una vez se haya producido la conversión A/D, tendremos el dato convertido en esta dirección, por lo que solo habrá que leerlo. A continuación se muestra la figura en la que aparecen las direcciones tanto de los registros de estado como de los registros de control.

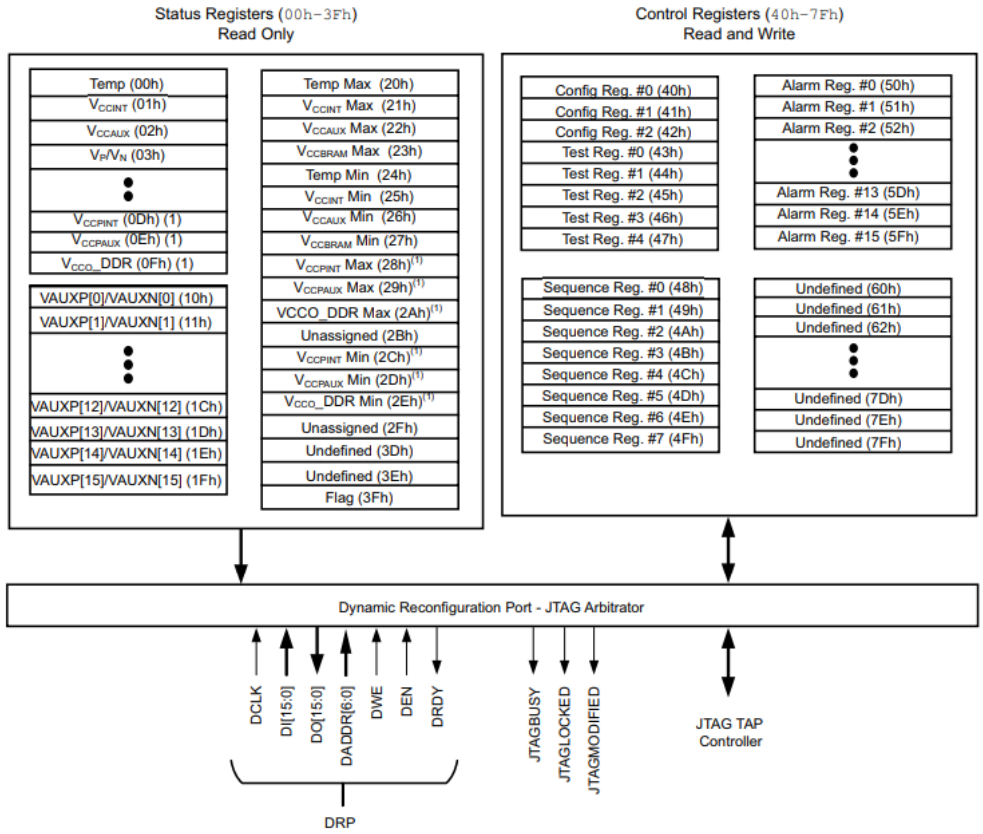

Figura 3.25: Interfaz de registros del XADC [18]

En nuestro caso escribiremos en el puerto **DADDR[6:0]** el valor x"1e", indicando el registro de estado que queremos leer. En este se encontrará el valor correspondiente a la conversión A/D perteneciente al canal **VAUXP14/VAUXN14**.

Funcionamiento del XADC

Una vez aclarada la configuración que hemos seleccionado para el XADC, se procede a explicar cómo se ha llevado a cabo el control del mismo a través de sus señales, las cuales nos aparecerán una vez instanciado el bloque IP. Los nombres de las señales que se van a usar para detallar el funcionamiento son genéricos, es decir, no se corresponden con los que aparecen dentro del fichero .vhd una vez instanciado el IP. Dicho esto comenzamos.

A pesar de que el XADC lo tenemos constantemente convirtiendo, los resultados de dicha conversión no se mostrarán por el puerto de salida a menos que se lo solicitemos. Para ello, lo que haremos una vez escrita en la señal **DADDR[6:0]** la dirección del registro que queremos leer, en nuestro caso la dirección del canal analógico auxiliar 14,

x“1e”, será asignar la señal **EOC** a la señal **DEN**. De este modo, cuando el XADC ponga el valor ’1’ en la señal **EOC** indicando que se ha finalizado una conversión, se pondrá a ’1’ la señal **DEN**, indicándole al XADC que se quiere hacer un acceso de lectura. En el momento en el cual el XADC coloque el dato en los 12 bits más significativos de la señal **DO[15:0]**, activará este mismo la señal **DRDY**, indicando que el dato está estable. Esta señal, además, como ya se mencionó, la usaremos para notificar a los bloques posteriores la existencia de un nuevo dato. Mediante este proceso conseguimos tener el XADC convirtiendo los datos que le entran por la entrada auxiliar número 14 de forma continua y controlamos cuando tenemos un dato nuevo convertido. A continuación se muestra una figura de cómo sería el proceso descrito y el tiempo de ejecución aproximado.

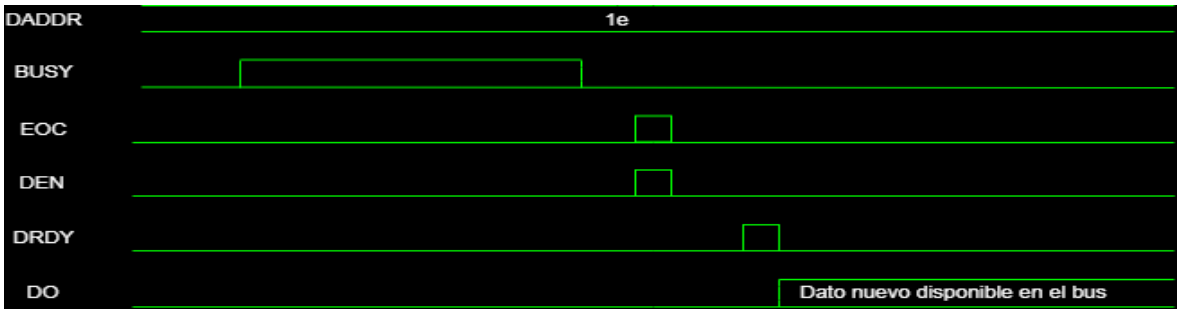

Figura 3.26: Proceso de conversión de un dato

Así mismo, en la figura anterior se ha incluido la señal **BUSY**, que, aunque en el código no se ha usado, sirve para entender mejor cómo es el proceso de conversión. Esta señal es puesta a ’1’ por el XADC cuando está realizando una conversión y colocada a ’0’ cuando esta finalice.

Media

Siguiendo con el análisis de los bloques del receptor nos encontramos con este, el cual aparece en la figura 3.3 con el nombre **MEDIA_MOVIL**. La función que realiza, como su nombre indica, es ir calculando el valor medio de un número determinado de muestras. El proceso mediante el cual se realiza el cálculo se basa en un vector de 64 posiciones, donde cada una de estas admite muestras de una longitud de 12 bits. Una vez el bloque anterior, **ADC**, tenga un nuevo dato convertido, pondrá la señal **FLAG_INTERM** a ’1’, indicándonos que tenemos un nuevo dato con el que calcular la media. Este será introducido en una nueva posición del vector de 64 muestras, usando para ello un contador con el fin de saber en qué posición nos encontramos. En el momento en que el vector esté completo, posiciones de 0-63 ocupadas, la muestra más antigua se encontrará en la posición que nos indique el contador, siendo en esta

en la que escribiremos la nueva. Una vez escrita, para realizar el cálculo de la media tendremos un acumulador, al que le restaremos la muestra que queda fuera de la ventana y sumaremos la nueva, para luego dividir el resultado entre la longitud total del vector, es decir, 64.

La longitud del vector corresponde al número de muestras equivalente a 2 bits, siendo la frecuencia de muestreo del conversor 77 KSPS y la tasa de transmisión del sistema 2400 b/s, como ya se mencionó en su momento. A continuación se muestra una figura para explicar el funcionamiento de la ventana móvil.

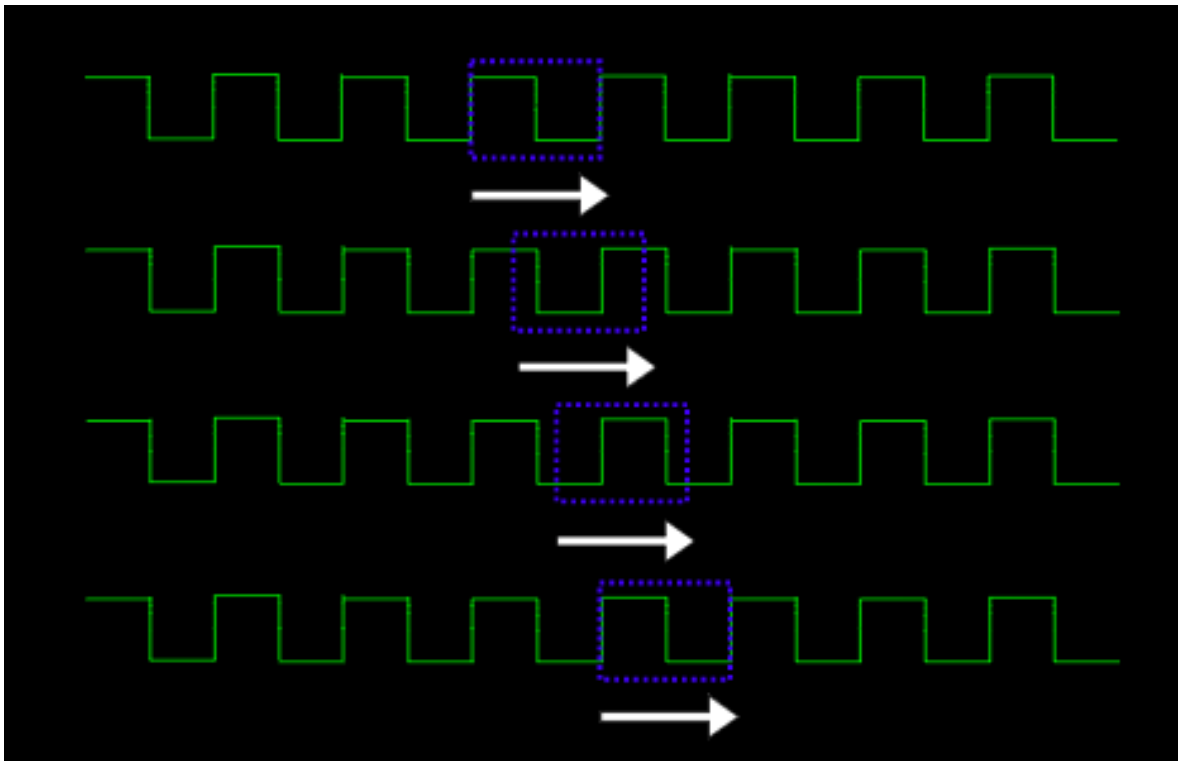

Figura 3.27: Desplazamiento de la ventana móvil con la línea de reposo del sistema

En la figura 3.27 aparece el proceso de avance de la ventana, destacar que el proceso de cálculo de la media para 2 bits se realizaría en 64 pasos, tantos como posiciones tiene el vector, mientras que en la figura se ha realizado en 4 con el fin de generar una idea global de cómo se hace. En la figura se puede observar el proceso descrito anteriormente, en el que al avanzar la ventana cogemos una nueva muestra por la derecha y desechamos la más antigua por la izquierda.

El valor de la media se irá calculando de forma continua, pasándose este de la misma forma al bloque correspondiente. Habrá ciertas ocasiones en las que el valor

de la media para unos determinados bits crezca o disminuya en exceso. Esto ocurrirá cuando haya dos '1' seguidos o el caso contrario, dos '0' seguidos. Aunque esto en un principio pueda parecer un problema, no lo es, ya que debido al uso de codificación Manchester cuando tengamos dos bits seguidos iguales, el siguiente será el opuesto. Así a los bloques encargados de tomar la decisión de si lo que han recibido se trata de un '1' o un '0', no les afectará el error en la media prácticamente.

Todo el proceso de cálculo se puede representar con la figura 3.28, siendo en nuestro caso **M=64**, **x(n)** la nueva muestra e **y(n)** la media calculada.

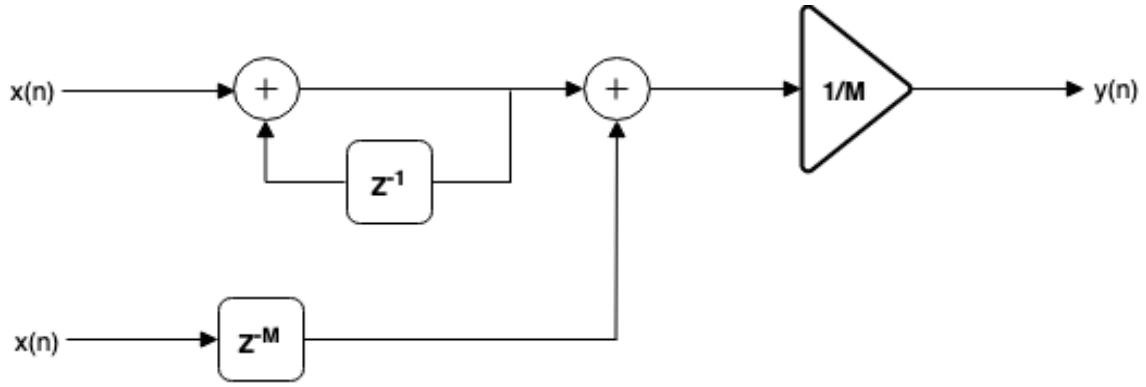

Figura 3.28: Diagrama de flujo para el cálculo de la media

A continuación se muestra la máquina de estados diseñada para implementar el diagrama de la figura 3.28.

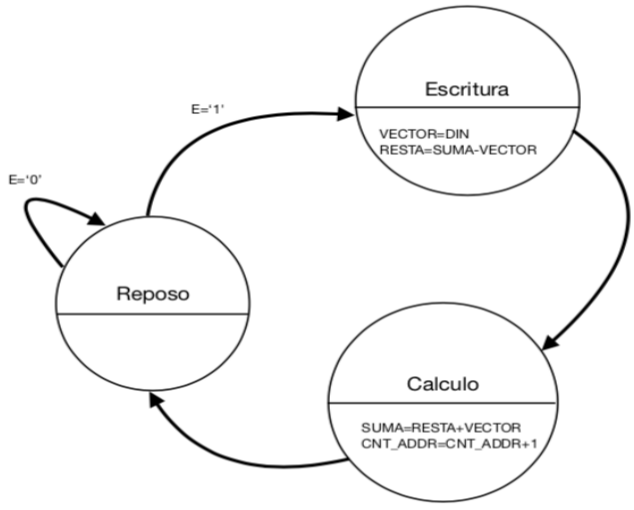

Figura 3.29: Máquina de estados encargada de realizar el cálculo de la media

Destacar la necesidad de añadirle a la señal encargada de calcular la suma $\log_2(\mathbf{M})$ muestras, mientras que para la realización de la división se descartarán los **M** bits

menos significativos del acumulado en la suma, asignándose el resultado a la señal **MEDIA_MOVIL**.

Comparador

Este bloque, situado en la figura 3.3 con el nombre **DECISOR_1**, como su nombre indica, se va a encargar de tomar una primera decisión, es decir, valorará a través de la señal que nos indica el valor de la media proporcionada a través del bloque anteriormente descrito, **MEDIA_INTERM**, si el valor de voltaje codificado en 12 bits por el conversor A/D, el cual es representado por la señal **SALIDA_INTERM_2**, se trata de un '1' o de un '0'.

Recordar de nuevo que por cada bit tendremos unas 32 muestras, por lo que tomaremos 32 decisiones aproximadamente. A continuación se mostrará un ejemplo para intentar conseguir una mayor comprensión por parte del lector. Supongamos un valor de voltaje codificado por el XADC en 12 bits, por ejemplo, "101101010010". Este valor será escrito en la señal **SALIDA_INTERM_2** por el XADC y recibido por el bloque actual como entrada. Una vez leído el valor, mediante la señal **MEDIA_INTERM**, la cual también es una entrada del bloque actual, se comprobará si el valor de voltaje es mayor o menor que la media, decidiendo así un '1' o un '0'. Una vez decidido habremos convertido una secuencia de 12 bits, en el ejemplo "101101010010", en '0' o '1'. Estos valores decididos no serán directamente los enviados por el emisor del sistema, dado que, como hemos mencionado en múltiples ocasiones, cada bit estará formado aproximadamente por 32 muestras, siendo las decisiones tomadas en este bloque parte de estas. A continuación se muestra una figura representativa del proceso descrito.

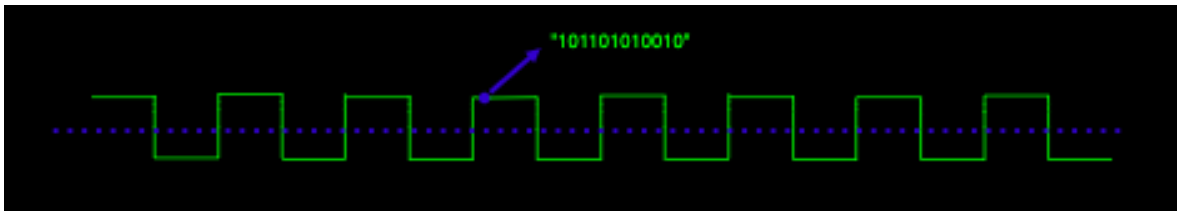

Figura 3.30: Ejemplificación de la decisión respecto a un valor de voltaje dado

En la figura se puede observar cómo la secuencia ejemplo mencionada, "101101010010", sería la representación de un valor de voltaje dentro de un bit. El sistema en este caso escribiría en la señal **DECIDIDO_INTERM_1** un '1', ya que el valor de voltaje es mayor que la media, línea azul punteada vista en la figura. Una vez decidido el valor, se pondría también a '1' la señal **VALID_INTERM**, indicando así

al bloque siguiente la existencia de un nuevo valor decidido. Igual que en los bloques descritos con anterioridad, las señales mencionadas se pueden observar en la figura 3.5, así como su interconexión.

Para este bloque no se ha considerado la representación de la máquina de estados desarrollada, dado que solo constaba de dos estados y no aportaría información extra al proceso descrito.

Decisor

Mediante este bloque, que se corresponde en la figura 3.3 con el nombre **DECISOR_2**, verificaremos si los '1' y '0' decididos por el bloque anterior son correctos o si ha habido alguna decisión errónea. Para ello se ha diseñado un algoritmo en el que, a través de la suma de las decisiones realizadas en el bloque anterior, dilucidamos si el emisor nos ha mandado un '1' o un '0'.

El proceso de decisión, como ya se ha mencionado, se basa en la suma de la señal recibida del bloque anterior, **DECIDIDO_INTERM_1**. Partiendo de la idea de que cada bit tendrá aproximadamente unas 32 muestras, se ha utilizado un valor de referencia por encima del cual la suma del valor decidido por el bloque anterior se puede considerar un '1' y en el caso contrario un '0'. En nuestro caso el valor concreto utilizado ha sido 20, así nos aseguramos un buen margen de error. De esta forma, aunque el valor de la media falle en algún caso como en los cambios de bit, donde puede haber valores muy cercanos a la media, además de los errores mencionados en la sección **MEDIA_MOVIL**, la decisión tomada por este bloque seguirá siendo fiable. A continuación se muestra una figura en la que se puede ver el problema en los cambios de bit.

Figura 3.31: Captura real de una prueba de transmisión realizada

La figura 3.31 es una captura real de una prueba de transmisión realizada en la que ya no se observan los pulsos puramente cuadrados representados hasta el momento. Ahora se puede observar cómo en los cambios de bit tenemos valores que habrá que decidir como '1' o como '0', pudiendo la elección alterar nuestra suma, de ahí la solución planteada del margen de error.

En el momento en el que dentro de este bloque la suma de bits decida dos '1' seguidos, tendremos el patrón de comienzo establecido por el emisor, y en consecuencia, un nuevo carácter recibido. Este bloque se encargará, además, una vez encontrado el patrón "11", de concatenar la secuencia de bits recibidos dándole forma al carácter enviado por el emisor. Esto se hará concatenando los bits recibidos de derecha a izquierda, posicionando el primer bit recibido, menos significativo del emisor, en la posición más significativa del nuevo vector. Una vez finalizada la concatenación, se escribirán los 16 bits correspondientes a esta en la señal **DECIDIDO_INTERM_2** para ser transmitidos al siguiente bloque, indicando, además, la disponibilidad de una nueva decisión mediante la puesta a '1' de la señal **HABILITACION_1**.

Del mismo modo que para el anterior bloque, **Decisor**, no se representará la máquina de estados diseñada, siendo los motivos para ello su gran complejidad debido a un alto anidamiento para las condiciones. En este caso se insta al lector a consultar la referencia en la que se encuentra el código correspondiente al proyecto si desea ver el proceso más en profundidad.

Decodificador

Seguidamente, mediante este bloque, el cual toma el mismo nombre en la figura 3.3, se realiza la decodificación de la serie recibida. Para explicar el proceso realizado, se hará uso de una figura en la que aparece un ejemplo concreto:

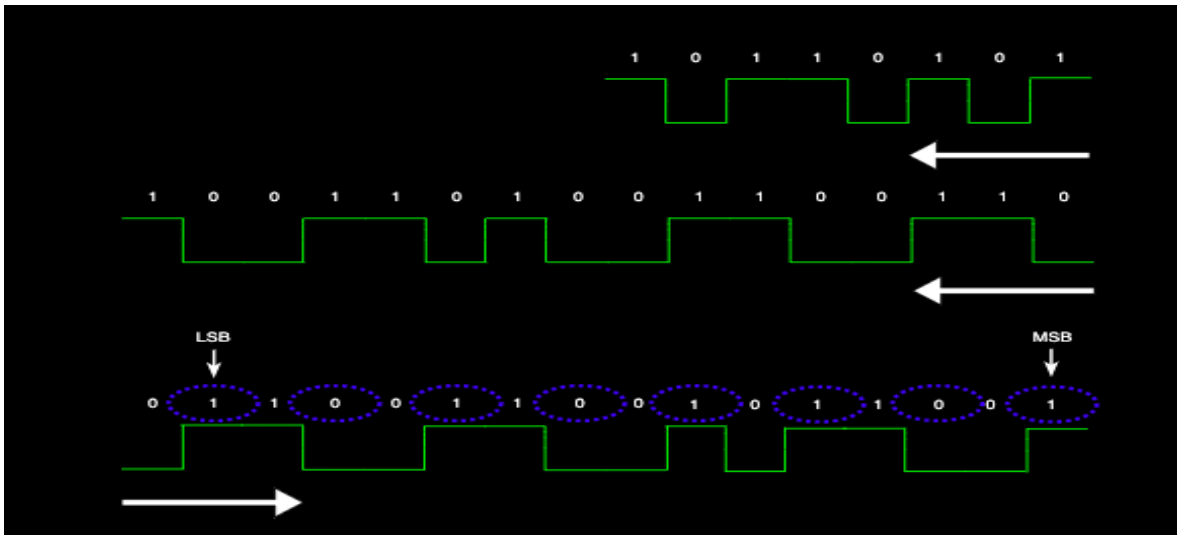

Figura 3.32: Ejemplo del proceso de decodificación

En la figura 3.32 se puede observar para el carácter correspondiente a la secuencia "1011010101" tanto el proceso de concatenación mencionado en el apartado anterior, en el cual, a la secuencia de 16 bits recibida se le daba la vuelta pasando el bit menos significativo a la posición de mayor importancia, como el proceso de decodificación. Este último, como se puede observar en la última forma de onda de la figura, se realiza seleccionando los bits de manera alterna, parte resaltada en azul. De este modo, el bit menos significativo de la secuencia decodificada será el que aparece en la figura redondeado en azul con la etiqueta de **LSB**, siguiendo para los siguientes el sentido marcado por la flecha.

Una vez concluida la decodificación del carácter recibido, se le indicará esta al siguiente bloque mediante la señal **HABILITACION_2**, siendo los 8 bits correspondientes al carácter escritos en **DECODIFICADO_INTERM**.

Para finalizar con este bloque, se muestra a continuación una figura representativa de la máquina de estados desarrollada, la cual, a pesar de su sencillez, puede ser clarificante para el lector.

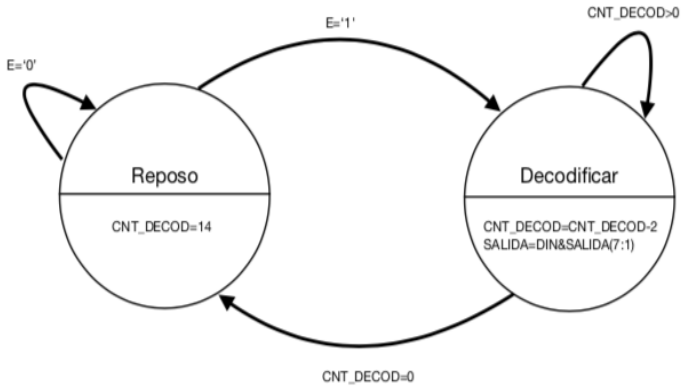

Figura 3.33: Máquina de estados encargada de controlar el proceso de decodificación

Transmisor puerto serie

Siendo este el último bloque perteneciente a la parte digital del sistema representado mediante la figura 3.22, tiene una gran similitud con el bloque **Recepción puerto serie** que aparece en la figura 3.10, ya que al igual que en este, se hará uso de un IP con el fin de realizar la comunicación puerto serie, tratándose en este caso de la transmisión en lugar de la recepción. Dentro de la figura 3.3 se puede encontrar con el nombre **AXI_TRANSMISOR**.

El asistente de configuración del IP se puede observar en la figura 3.11, correspondiente a la recepción puerto serie, puesto que, al ser el mismo IP, el asistente también es el mismo. Lo que sí que se ha cambiado son los campos que aparecen, modificándose tanto la opción **BAUD RATE**, la cual se ha puesto a su máximo valor, 921600 b/s, y la frecuencia de reloj, 300 MHz, dado que si no subíamos esta, no podíamos transmitir a esta tasa tan alta. En cuanto al reloj, como se verá posteriormente, es generado a través del bloque **DCM** observable en la figura 3.3.

En lo que se refiere al proceso de configuración de la transmisión puerto serie, este es similar al descrito en el bloque **Recepción puerto serie**, pero interviniendo diferentes señales. En la figura 3.34 se muestran las señales referentes al proceso de transmisión puerto serie y la forma en que lo hacen.

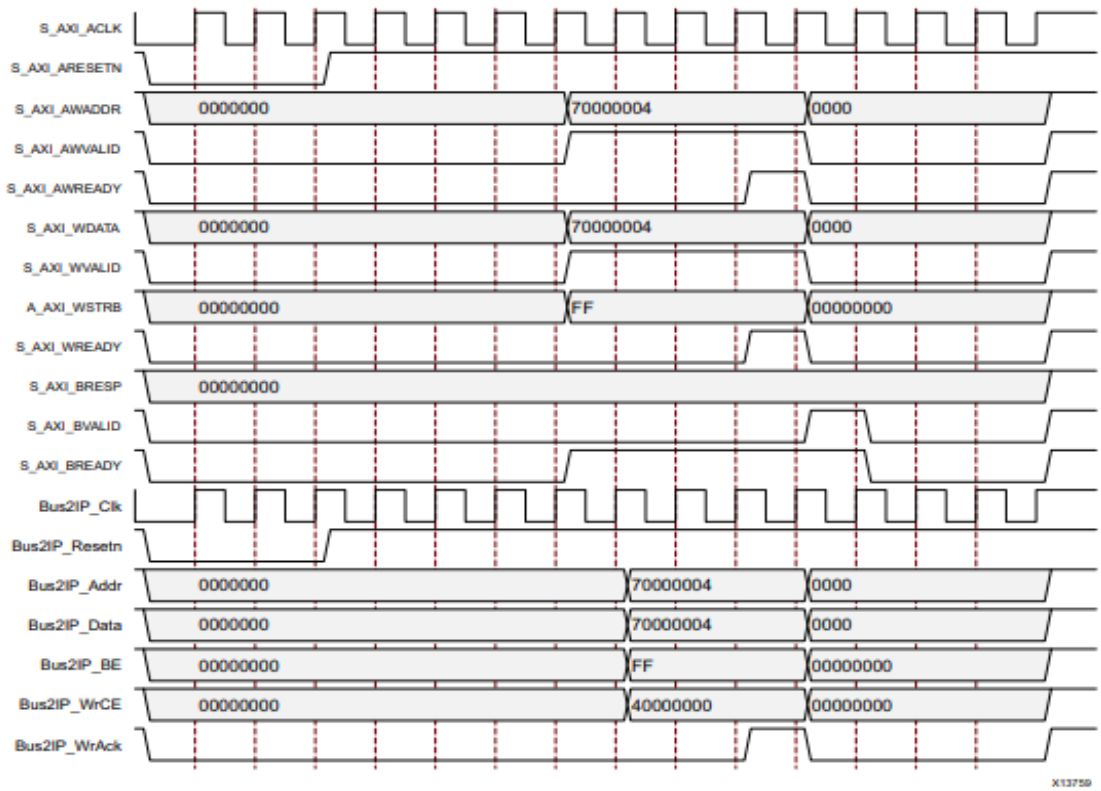

Figura 3.34: Operación de escritura del AXI [19]

Igual que se hizo en el bloque correspondiente a la recepción puerto serie, se va a proceder a realizar la explicación general del proceso transmisión puerto serie, para después ahondar en el uso que le hemos dado en nuestro Trabajo Fin de Grado.

1. El maestro escribe una dirección y un dato en las señales **S_AXI_AWADDR** y **S_AXI_WDATA** respectivamente, poniendo en ese instante a '1' tanto la señal **S_AXI_AWVALID** como **S_AXI_WVALID**. Así el maestro consigue dar validez a la dirección y al dato escritos. En el momento en que el maestro esté preparado para recibir una respuesta, pondrá también a '1' la señal **S_AXI_BREADY**.

2. El esclavo pondrá a '1' tanto la señal **S_AXI_AWREADY** como **S_AXI_WREADY**.

3. En el momento en que las señales **S_AXI_AWVALID**, **S_AXI_WVALID**, **S_AXI_AWREADY** y **S_AXI_WREADY** están en alto, el esclavo se guarda la dirección y el dato, pudiéndose poner a '0' en este momento todas estas.

4. El esclavo pondrá a '1' **S_AXI_BVALID** en el momento que haya una respuesta válida, produciéndose la transacción en el siguiente ciclo de reloj.

Todo este proceso se ha obtenido como ya se mencionó en la recepción puerto serie de la página de la referencia [20]. Respecto a nuestro caso particular, se han usado señales correspondientes al proceso de lectura, descrito en la sección 3.2.1, con el fin de conocer si la cola de transmisión está llena. Esto se realiza leyendo la posición 3 del vector relativo al registro de estado, cuya dirección en hexadecimal se puede observar en la figura 3.13. En el momento en el que el registro de datos en esta posición tome valor '1', deberemos dejar de escribir por el puerto serie, de lo contrario perderemos datos, dado que desbordaríamos la cola de transmisión.

Este proceso se realizará cuando tengamos en el bloque anterior un nuevo carácter decodificado, este instante será indicado por la señal **HABILITACION_2**. En la señal **DECODIFICADO_INTERM**, tendremos los 8 bits correspondientes al carácter que debemos mandar por el puerto serie.

Destacar por último, antes de comenzar con el siguiente bloque, la utilidad de la señal **S_AXI_BRESP**, ya que nos indica si la transmisión puerto serie se ha realizado de manera correcta o no. En la figura 3.35 se muestran los diferentes valores que puede tomar, indicándonos que todo se ha realizado de forma correcta mediante la respuesta "00".

BRESP[1:0]	Response
0b00	OKAY
0b01	EXOKAY
0b10	SLVERR
0b11	DECERR

Figura 3.35: Códigos de respuesta del AXI4-lite [22]

Estas respuestas se pueden ver también en el proceso de recepción puerto serie mediante la señal **S_AXI_RRESP**. En el apartado correspondiente no se mencionó debido a que en este Trabajo Fin de Grado no se ha utilizado esta señal, mientras que en la transmisión puerto serie, estas respuestas se usaron para realizar pruebas de funcionamiento.

Para intentar dar una explicación más clara acerca de las señales que hemos usado en este proceso, se muestra a continuación la máquina de estados desarrollada para realizar el control de este IP.

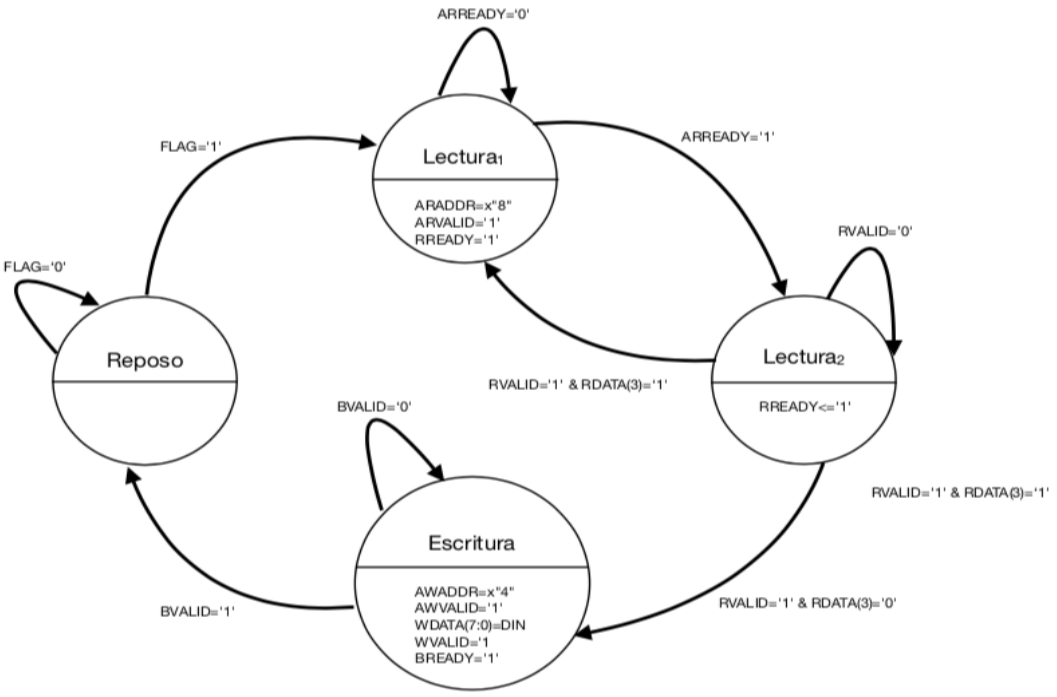

Figura 3.36: Máquina de estados encargada de controlar el IP usado en la transmisión puerto serie

Generador de reloj y reset

Mediante este bloque, como su nombre indica, generamos las señales de reloj y reset usadas en nuestro sistema, siendo estas comunes tanto a receptor como a emisor, por lo que perfectamente podrían haber sido explicadas en la sección correspondiente a este. El bloque puede ser observado no solo en la figura 3.2 con el nombre de **DCM**, sino también en la figura 3.3, pudiendo ver sus señales tanto de salida como de entrada, observando que no son las mismas a pesar de ser un bloque común. Esto se debe a una simplificación meramente de la figura. Del mismo modo que en bloques anteriores como el **ADC** o el **Transmisor puerto serie**, haremos uso de un IP para poder utilizarlo. El nombre de este es **Clocking Wizard**, cuyo asistente de configuración se puede ver en la figura 3.37.

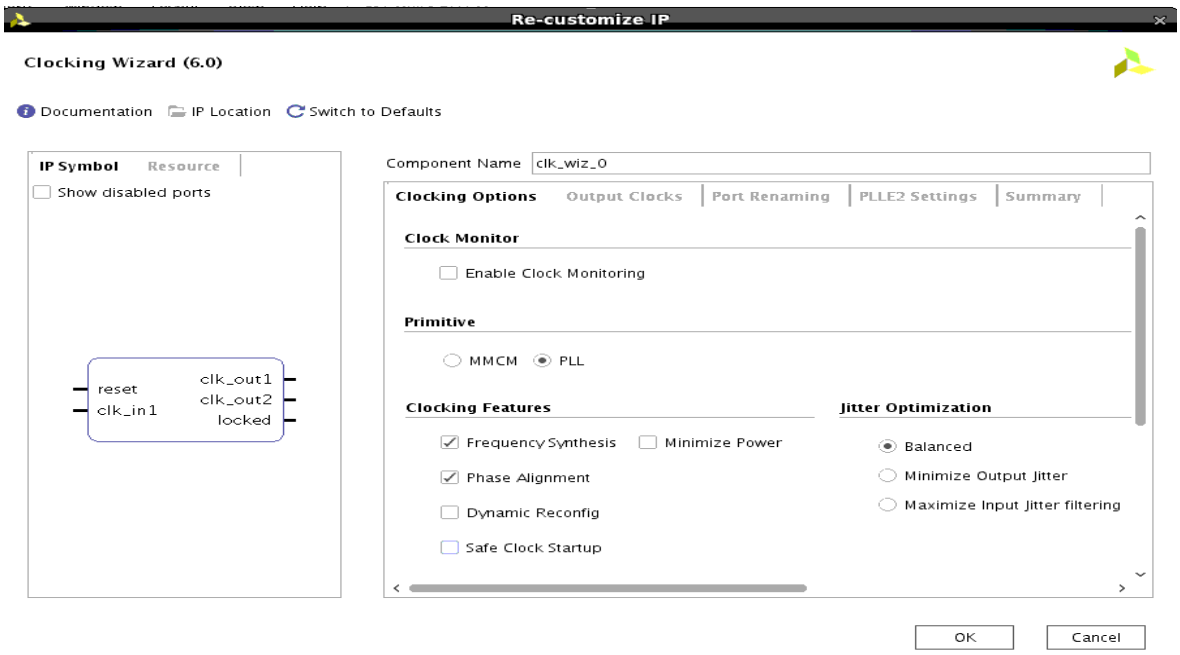

Figura 3.37: Asistente de configuración del IP Clocking Wizard

Este bloque se ha usado con la finalidad de generar varios relojes, con frecuencias de 300 y 100 MHz respectivamente. En la figura 3.2 solo se puede observar la utilización del reloj de 100 MHz, ya que el de 300 MHz se usa en el receptor, 3.3, y como medida de simplificación no se ha incluido en el esquema de la figura 3.2. La otra señal que nos es de gran utilidad y que nos la genera este IP es **Locked**, la cual nos indica cuándo los dos relojes generados están sincronizados. Esta señal en este Trabajo Fin de Grado la usaremos como reset del sistema completo, asegurándonos la no entrada de nuestros biestables en estado metaestable al pulsar el botón de reset. La señal tomará valor '1' cuando los dos relojes estén sincronizados, por lo que haremos que todo el sistema esté reseteado mientras esta señal valga '0'. En la figura 3.10 se muestra lo explicado,

viendo como **Locked** es conectado al reset de los demás bloques.

Respecto a la configuración de este IP dentro de nuestro proyecto, se observan en la figura 3.37 las diferentes opciones que nos da este. A continuación se enumerarán y se dará una breve explicación acerca de ellas y de las opciones seleccionadas:

- **Clocking Options**
- **Output Clocks**
- **Port Renaming**
- **PLLE2 Settings**
- **Summary**

Lo primero destacable dentro de la pestaña **Clocking Options** dentro de la opción **Primitive** es la selección de **PLL**, la cual contiene un subconjunto de funciones de **MMCM** [23], que serán más que suficientes en el desarrollo de nuestro proyecto. En la siguiente pestaña, **Clocking Features**, se han seleccionado dos opciones, **Frequency Synthesis**, con la que permitimos generar relojes con distinta frecuencia que el de entrada, que será el de 100 MHz generado por la FPGA, y **Phase Alignment**, con lo que conseguimos tener en fase los relojes de salida y el de entrada. Las otras opciones de esta pestaña han permanecido por defecto como venían en el asistente, quedándose dentro de **Jitter Optimization** la opción **Balanced** marcada, haciendo que el software elija el ancho de banda correcto para la optimización del *jitter*. Por último, dentro de la opción **Input Clock Information** debemos indicar la frecuencia del reloj de entrada al sistema, que como se ha mencionado antes, será la del reloj generado por la FPGA, 100 MHz.

La siguiente pestaña que debemos configurar es la que aparece como **Output Clocks**, donde en una ventana como la de la figura 3.38 deberemos indicar tanto el número de relojes de salida que queremos como las frecuencias a las que estos serán generados.

Output Clock	Port Name	Output Freq (MHz)		Phase (degrees)		Duty Cycle (%)	
		Requested	Actual	Requested	Actual	Requested	Actual
☑ clk_out1	clk_out1	300.000	300.000	0.000	0.000	50.000	50.0
☑ clk_out2	clk_out2	100.000	100.000	0.000	0.000	50.000	50.0

Figura 3.38: Configuración de los relojes de salida del IP Clocking Wizard

Las demás opciones no han sido modificadas, por lo que no se va a explicar nada más acerca de estas. Mencionar que la pestaña **Port Renaming**, como su nombre sugiere, permite modificar el nombre de la señal que nos indica el enganche de fase, que como se ha dicho, no se ha modificado, de ahí el nombre de **Locked**.

3.4. Canal

Por último y para finalizar el capítulo de una forma muy breve, se aborda el canal de nuestro sistema, el cual, como en todas las comunicaciones por luz visible, será el aire, siendo la luz procedente de bombillas o del sol un gran inconveniente como se mostrará en la parte experimental de este trabajo mediante ejemplos de evaluación de nuestro propio diseño.

Capítulo 4

Parte experimental

A lo largo de este capítulo se tratará de enumerar las pruebas seguidas durante el desarrollo del Trabajo Fin de Grado con el fin de conseguir el objetivo último, la obtención de un sistema VLC funcional.

4.1. Pasos seguidos durante el desarrollo del sistema

El primer bloque sobre el que iniciamos la construcción de nuestro sistema de comunicaciones se trata del **XADC**, el cual una vez configurado se simuló su funcionamiento mediante una sinusoide. Para ello se tuvo que configurar un fichero de texto como el que se muestra a continuación.

TIME	VAUXP[14]	VAUXN[14]
0	0.1	0
12987	0.16	0
25974	0.16	0
38961	0.1	0
51948	0.14	0
64935	0.14	0
77922	0.16	0
90909	0.14	0
103896	0.14	0
116883	0.16	0
129870	0.2	0
142857	0.04	0
155844	0.14	0
168831	0.14	0
181818	0.1	0
194805	0.1	0
207792	0.24	0
220779	0.16	0
233766	0.24	0
246753	0.16	0
259740	0.14	0
272727	0.1	0
285714	0.2	0
298701	0.16	0

Figura 4.1: Fichero de texto utilizado para la simulación del XADC

Como podemos observar en la figura, el fichero que aparece tiene 3 columnas, **TIME**, **VAUXP[14]** y **VAUXN[14]**, pudiendo añadir las que quisiésemos. Destacar que la unidad de medida de la columna referente al tiempo de simulación son nanosegundos, mientras que las otras representan valores de voltaje.

El caso que aparece en la figura 4.1 representa parte de una de las señales formada por pulsos cuadrados que hemos usado en simulación. En esta le damos valor solo a la entrada auxiliar de la FPGA **VAUXP[14]**, conectando a tierra la otra, como ya se mencionó en la parte correspondiente a la configuración del XADC del capítulo 3.

En el momento en que se comprobó el correcto funcionamiento del XADC, se empezó a trabajar en los IP correspondientes a la recepción puerto serie en primer lugar, y transmisión puerto serie a continuación. Se realizaron pruebas de funcionamiento de ambos a través de simulación, juntándose en un proyecto, una vez concluidos, el XADC con el bloque encargado de efectuar la transmisión puerto serie.

Una vez juntos, se desarrolló otra prueba de simulación para después elaborar la primera prueba experimental del proyecto, en la que mediante el generador de señales se creó una sinusoide cuyo valor de voltaje estaba acotado entre 0-1 voltio para no dañar la FPGA. La señal se introdujo al XADC y se visualizó el puerto serie desde el PC mediante el programa **SerialPlot**. Con esto se comprobó que los IP relativos funcionaban correctamente, pudiendo proseguir con el desarrollo del proyecto.

Llegados a este punto, se diseñó la etapa analógica relativa al emisor, la cual podemos observar en la figura 3.4, y una primera parte de la etapa analógica correspondiente al receptor, que incluía el fotodiodo y el amplificador de transimpedancia. Con estas etapas se realizaron pruebas con pulsos cuadrados de amplitud 0 a 3,3 voltios de frecuencia 1,2 kHz con el fin de imitar lo máximo posible los pulsos generados por la FPGA. Así se ajustaron de forma experimental los valores de condensadores y resistencias utilizados en las etapas diseñadas, además de buscar el valor de alimentación necesario para que el diodo luciese correctamente. A continuación se muestra una imagen del laboratorio, de la configuración de la fuente para alimentar a ±7 voltios el circuito, $\pm\mathbf{V_{cc}}$ respectivamente.

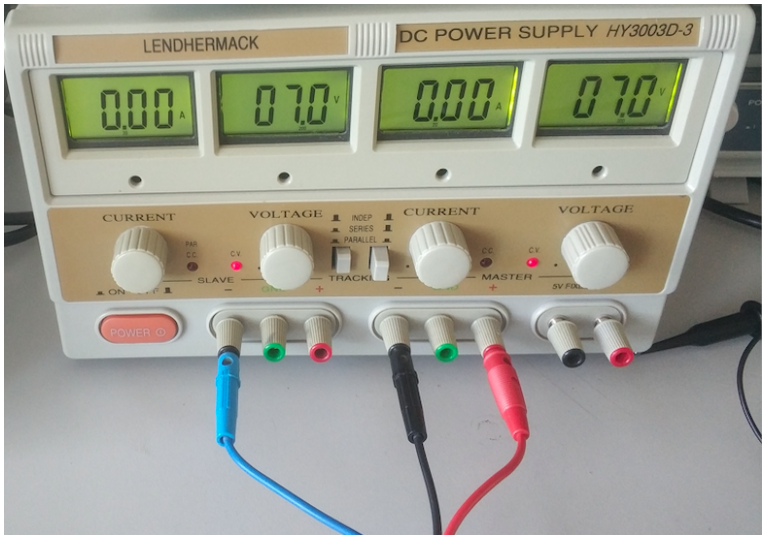

Figura 4.2: Alimentación de nuestro sistema

En la imagen se puede observar la fuente de alimentación trabajando en modo serie, mediante la selección del botón izquierdo de la opción **TRACKING**. De esta forma se consigue tener $-\mathbf{V_{cc}}$ en el cable azul, el negro como **GND** y $+\mathbf{V_{cc}}$ en el rojo.

Concluida la etapa analógica básica, se creó un proyecto nuevo en VHDL, en el que a la recepción puerto serie se le añadieron los bloques encargados de realizar la codificación Manchester y la transmisión a 2400 b/s. Una vez terminado se simuló mediante un sencillo *test bench* y verificó su funcionamiento. Con este bloque teníamos completado el control del emisor. Los bloques que lo formaban son los que aparecen en la figura 3.10 salvo el **DCM**, ya que para hacer pruebas usábamos el reloj de 100 MHz generado por la FPGA, y **SIETE_SEG**, último bloque añadido en el proyecto.

Llegados a este punto, se modificó la parte analógica correspondiente al receptor con el fin de ajustar los valores de amplitud de los pulsos. Una vez acotados entre 0-1 voltio, pudiendo así introducírselos al conversor de la FPGA. Se unió en un proyecto la parte correspondiente a la recepción puerto serie, transmisión y codificación de pulsos con el XADC y la transmisión puerto serie. Con ello podríamos probar a mandar caracteres mediante un PC por un puerto y recibirlos en otro. A continuación se muestran capturas tomadas con la ayuda del osciloscopio de una serie de caracteres transmitidos por el PC emisor.

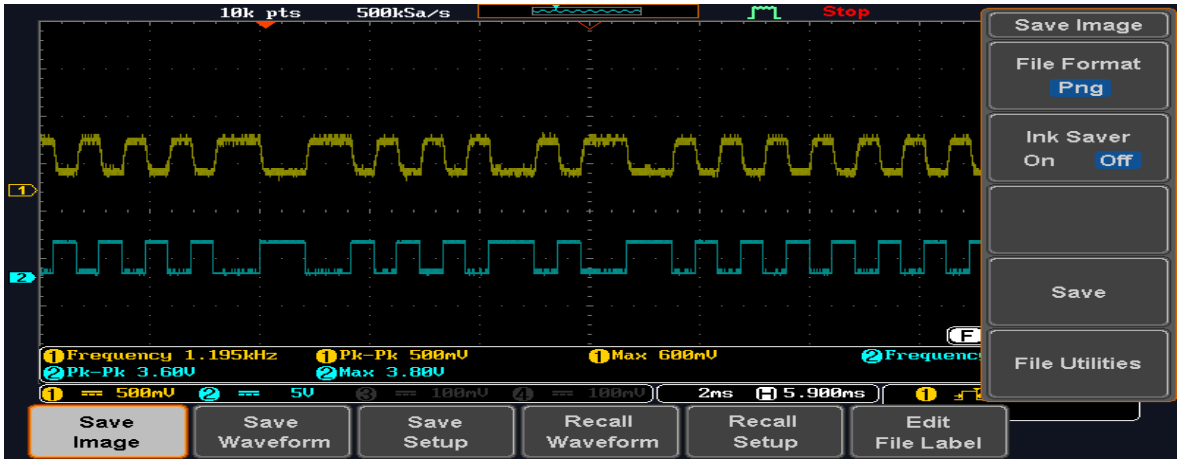

Figura 4.3: Señales correspondientes al carácter a codificado

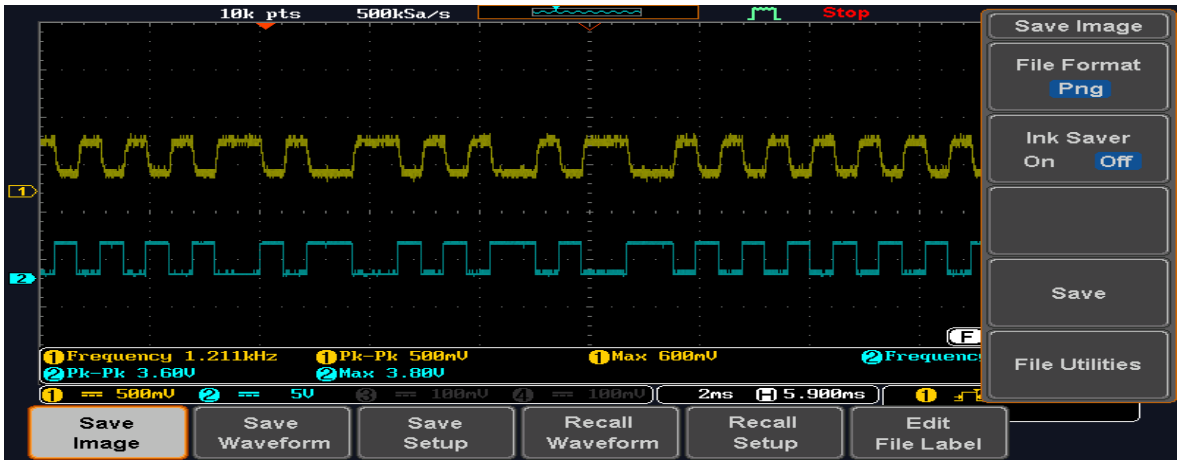

Figura 4.4: Señales correspondientes al carácter b codificado

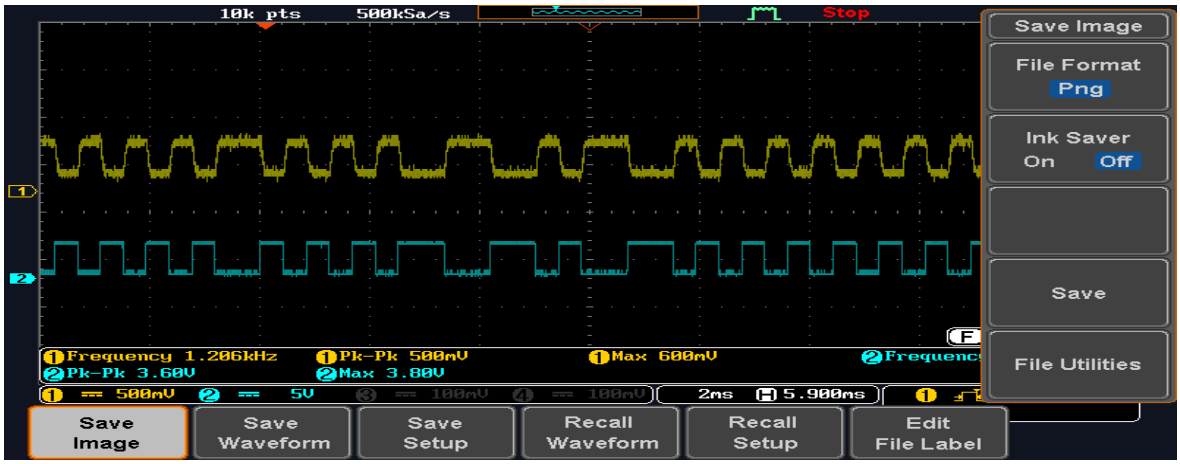

Figura 4.5: Señales correspondientes al carácter h codificado

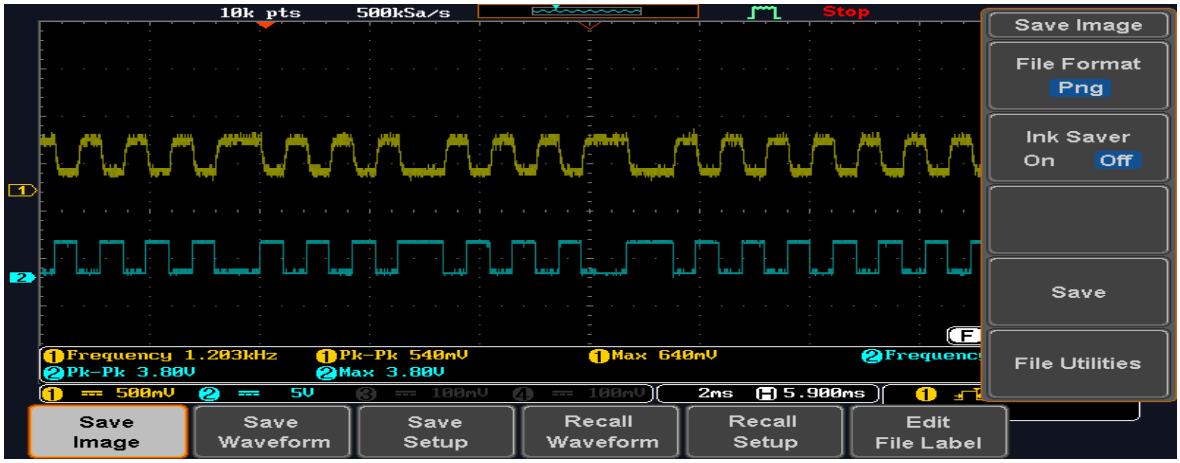

Figura 4.6: Señales correspondientes al carácter x codificado

En las imágenes se puede observar en azul el canal 2 del osciloscopio, que representa la señal transmitida por el emisor. Como ya se mencionó en su momento, el bloque **TX**, encargado de transmitir los bits, los niega mediante una puerta **not**, ya que el circuito receptor invierte la señal. La señal del canal 1, amarilla, es tomada a la entrada del conversor A/D de la FPGA, pudiéndose observar en cada una de las figuras el patrón "11" que indica el comienzo de un carácter codificado.

Para realizar estas capturas, se ha utilizado la función del osciloscopio de disparo mediante **Pulse Width**, configurándola para que cuando la anchura de un pulso sea mayor a 416 μs, $\frac{1}{2400}$ s, se produzca la captura de la secuencia recibida. Esto se producirá cuando nos llegue el patrón indicador de una nueva secuencia, "11", dado que el ancho de pulso será el doble que el configurado para el disparo del osciloscopio.

Una vez se concluyó todo el bloque digital perteneciente al receptor, para su simulación se utilizó una función del osciloscopio con la que se puede guardar la señal en un fichero .csv. Se creó un script de MATLAB con el fin de leer el archivo y realizar un submuestreo, pasando de la frecuencia a la que muestrea la señal del osciloscopio a 77 KSPS, frecuencia de muestreo configurada para el XADC. Continuando con el script de MATLAB, el cual se puede encontrar en la dirección correspondiente a la referencia [24] con el nombre **Senyal.m**, cabe resaltar que una vez realizado el proceso de submuestreo se modificaban los valores de amplitud leídos del fichero, con el fin de representar los voltajes reales, y se escribían en el archivo del que realiza el conversor XADC la lectura de valores, el cual se representa en la figura 4.1.

En el momento en el cual se concluyeron tanto el receptor como el emisor, se utilizaron caracteres reales capturados mediante el osciloscopio para simular su

comportamiento dentro del receptor del sistema. Una vez se consiguió el funcionamiento correcto de esta etapa, se unieron en un único proyecto tanto emisor como receptor, con el fin de solo usar una FPGA en nuestro sistema. Además se añadió al sistema el bloque **SIETE_SEG**, que da una mayor visibilidad a lo que estamos haciendo, al mostrar los números recibidos por el puerto serie en el visualizador de 7 segmentos de la FPGA.

Por último se introdujo la etapa encargada de controlar el voltaje de entrada a la FPGA, representada en la figura 3.9. El funcionamiento global del sistema no se simuló, ya que una vez teníamos funcionando por separado tanto emisor como receptor, solo había que unir bloques. De este modo, el funcionamiento global se realizó de forma experimental como se detalla en la siguiente sección.

4.2. Pruebas realizadas en el sistema VLC desarrollado

Una vez montado el sistema VLC, para probar su funcionamiento se utiliza el programa **PuTTy** en el PC. Con este se mandan caracteres desde un puerto de comunicaciones a una velocidad, 2400 b/s, recibiéndolos por otro a 921600 b/s. En el programa activamos la opción de eco, pudiendo ver en el terminal lo que mandamos por un puerto, y lo que recibimos por el otro. A continuación se muestra una imagen representativa de lo descrito, en la que la ventana del terminal de la izquierda es lo transmitido y la ventana de la derecha lo recibido.

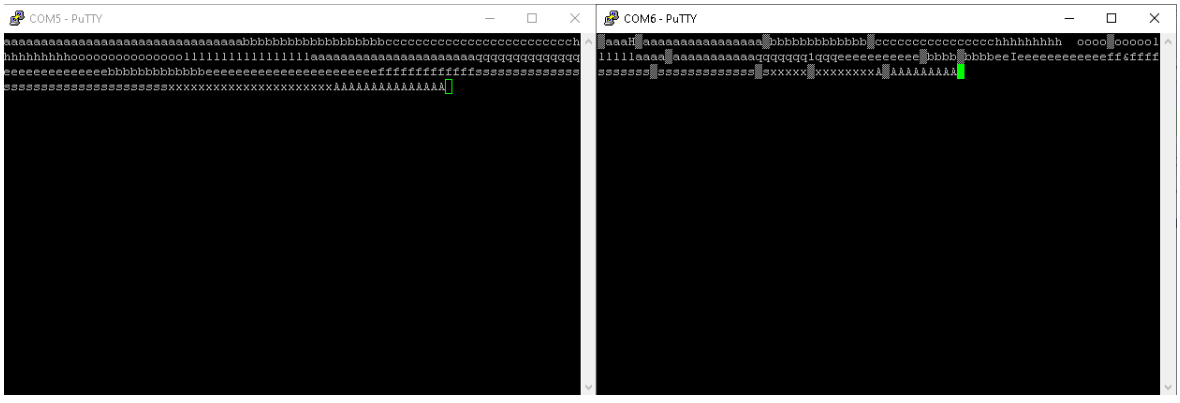

Figura 4.7: Serie de caracteres enviados y recibidos

Podemos observar cómo lo transmitido es recibido en su mayoría, corroborando así el correcto funcionamiento del sistema de forma experimental. Más adelante se mostrarán una serie de pruebas que han sido llevadas a cabo con la finalidad de verificar de forma más precisa el rendimiento del nuestro sistema. Destacar que el funcionamiento del sistema, como ya se mencionó en el capítulo 3 en la sección

correspondiente al canal, depende de factores externos como la iluminación de la sala. Destacar la necesidad de uso de un componente externo para la transmisión puerto serie. Este es el **CP2102**, se trata de un puente entre el USB y el Transmisor-Receptor Asíncrono Universal UART (Universal Asynchronous Receiver-Transmitter) con el cual se puede realizar la comunicación de manera sencilla entre la FPGA y el PC. La FPGA no puede recibir por el puerto serie a una velocidad (2400 b/s) y transmitir a otra (921600 b/s), por lo que se ha hecho uso de este. Podemos observarlo en el capítulo 3 en la figura 3.3 con el nombre de **CP2102**.

Tras esta breve aclaración, se procede a retomar la línea de pruebas realizadas. A continuación se muestran una serie de capturas tomadas a la entrada de la FPGA con distintas situaciones de iluminación.

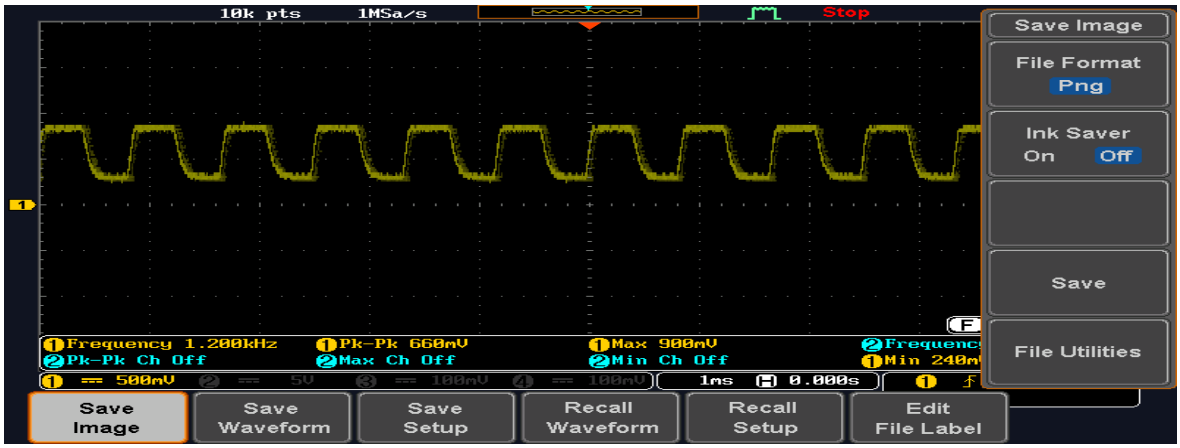

Figura 4.8: Señal recibida a la entrada de la FPGA con luminosidad baja

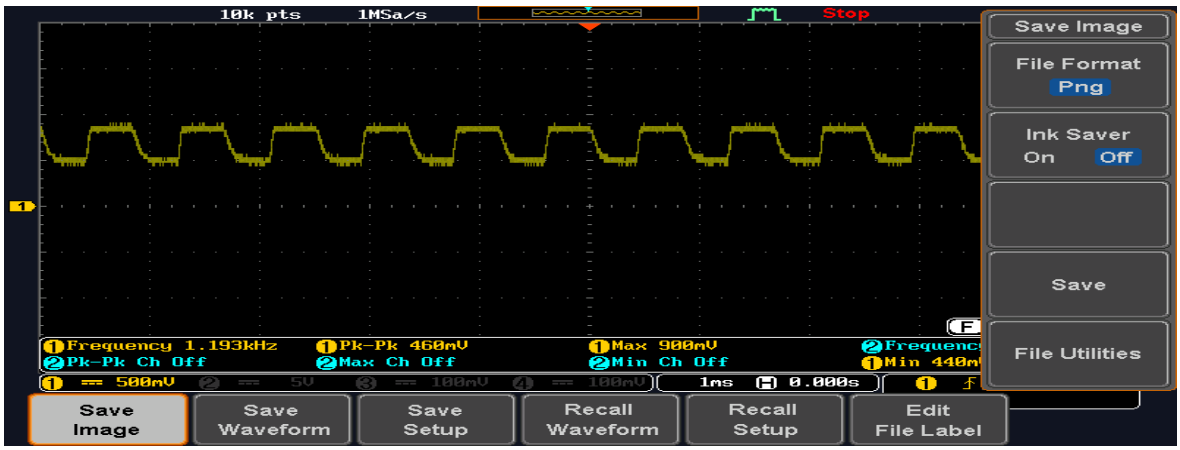

Figura 4.9: Señal recibida a la entrada de la FPGA con luminosidad media

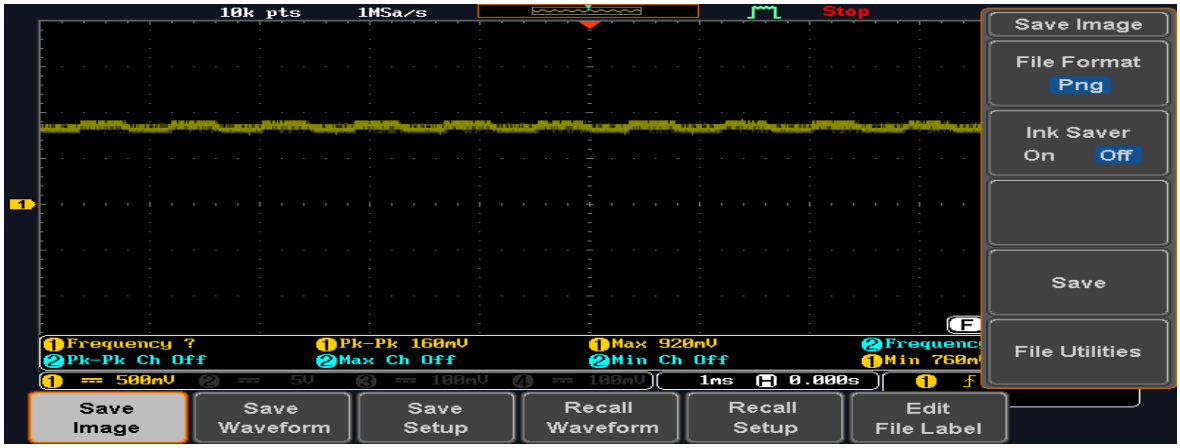

Figura 4.10: Señal recibida a la entrada de la FPGA con luminosidad alta

La figura 4.8 se ha tomado tapando tanto el emisor como el receptor con la mano, por lo que la luz incidente era bastante baja. Seguidamente se realizó la captura de la figura 4.9 en la que teníamos las persianas bajadas del laboratorio, por lo que no había una gran luz en la sala. Y por último la figura 4.10 se tomó en el mismo laboratorio, pero, en este caso, con una gran luz incidente, ya que todas las persianas estaban abiertas.

De esta forma, se ha podido ver cómo afecta la luz a nuestro sistema, pudiéndose observar a simple vista en las figuras anteriores la aparición de un *offset* en la señal cuanta mayor luz incidente tenemos. A continuación se muestra una tabla comparativa de las 3 figuras anteriores, en la que podemos ver lo descrito.

Figura	$\mathbf{V_{max}}$	$\mathbf{V_{min}}$	$\mathbf{V_{offset}}$
4.8	990 mV	240 mV	0 mV
4.9	990 mV	440 mV	200 mV
4.10	920 mV	760 mV	520 mV

Tabla 4.1: Voltajes obtenidos para diferentes escenarios basados en luminosidad

En la tabla podemos observar cómo el *offset* va creciendo con el aumento de la luz incidente, quedándonos prácticamente sin señal en la figura 4.10.

El siguiente y último paso dentro del proyecto fue el desarrollo de una PCB tanto para la parte analógica correspondiente al emisor, como para la del receptor (ver **Anexo D**). A continuación se realizaron una serie de pruebas con el fin de ver cómo afectaba la distancia entre emisor y receptor a la señal. Seguidamente se muestran una serie de figuras en las que se han capturado los valores de voltaje de la señal que tendríamos a la entrada de la FPGA para diferentes distancias.

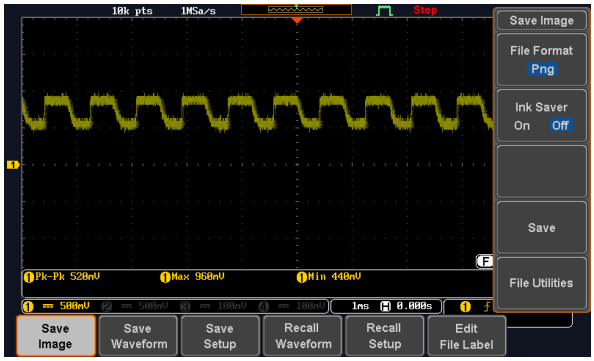

Figura 4.11: Emisor y receptor juntos

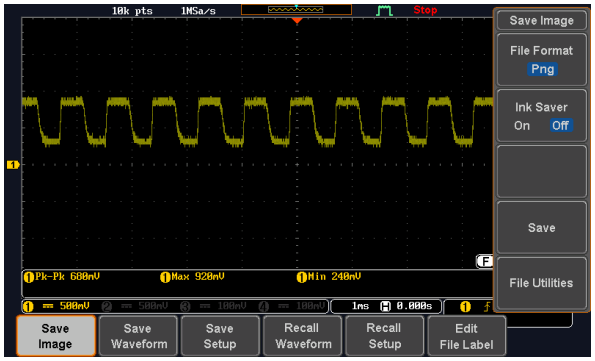

Figura 4.12: Emisor y receptor a 0,5 cm

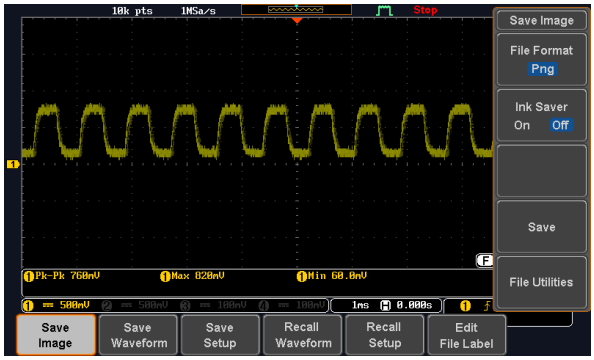

Figura 4.13: Emisor y receptor a 1 cm

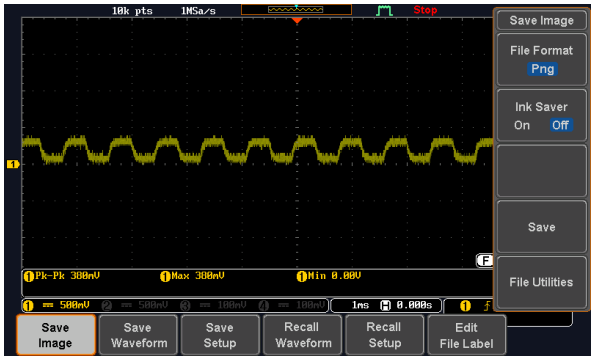

Figura 4.14: Emisor y receptor a 2 cm

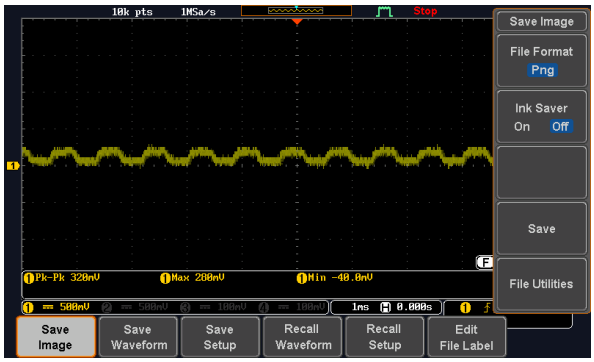

Figura 4.15: Emisor y receptor a 3 cm

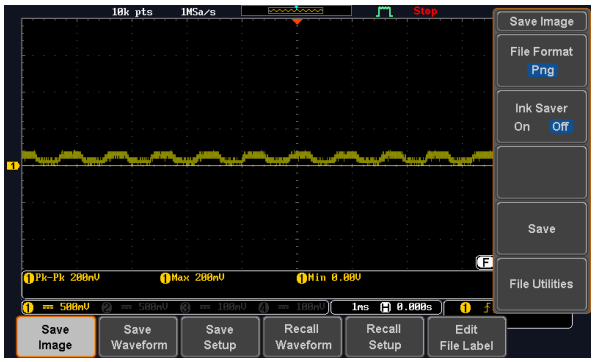

Figura 4.16: Emisor y receptor a 4 cm

Destacar la figura 4.11, en la que se puede observar la aparición de un *offset* en la señal cuando emisor y receptor se encuentran muy cerca. Seguidamente, se aprecia una disminución en la amplitud de la señal conforme aumentamos la separación entre ambos, llegándose prácticamente a no distinguir los pulsos en la figura 4.16, por lo que nuestro sistema no será capaz de discernir si la señal se tratá de un '1' o un '0'. A continuación se muestra una tabla en la que se aportan algunas distancias adicionales a las representadas en las figuras, pudiéndose ver de una forma más detallada cómo se va viendo modificada la señal recibida.

$\mathbf{D_{emisor-receptor}}$	$\mathbf{V_{max}}$	$\mathbf{V_{min}}$
0 cm	960 mV	440 mV
0,5 cm	920 mV	240 mV
1 cm	820 mV	60 mV
1,5 cm	480 mV	40 mV
2 cm	380 mV	0 mV
2,5 cm	300 mV	-40 mV
3 cm	280 mV	-40 mV
3,5 cm	240 mV	-40 mV
4 cm	200 mV	0 mV

Tabla 4.2: Voltajes obtenidos para diferentes escenarios basados en distancia

En la figura 4.17 se muestra una captura de la señal que introdujimos al conversor de la FPGA para una distancia de 6,5 cm entre emisor y receptor, momento en el cual se ha comprobado que nuestro sistema ya no es capaz de decidir de forma aceptable el carácter recibido debido a la baja amplitud de la señal.

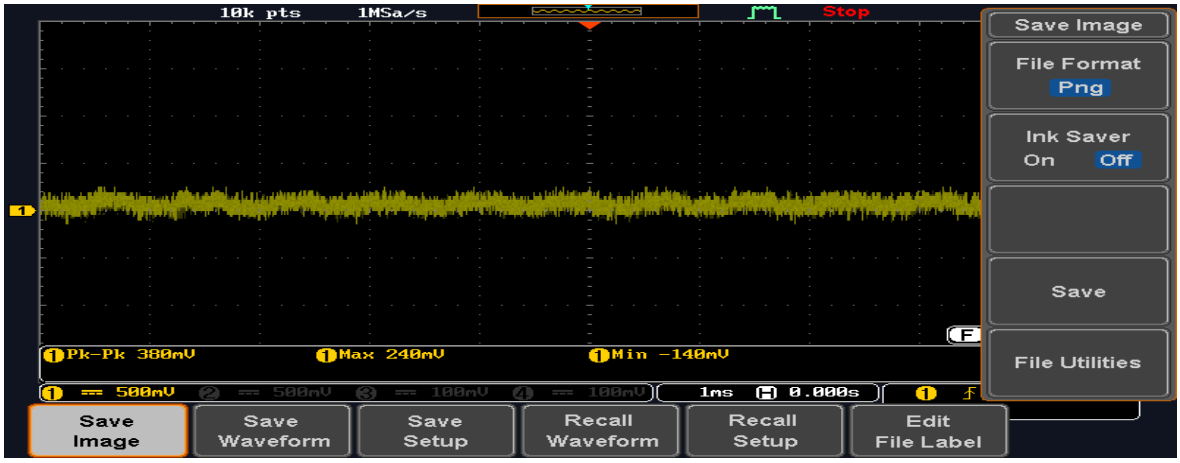

Figura 4.17: Señal recibida a la entrada de la FPGA a una distancia de 6,5 cm entre emisor y receptor

Cómo complemento se adjunta una tabla realizada con la que se puede observar cómo emisor y receptor no tienen por qué estar enfrentados uno y otro, sino que puede haber cierto ángulo entre ellos.

$\mathbf{Ángulo_{emisor-receptor}}$	$\mathbf{V_{max}}$	$\mathbf{V_{min}}$
0º	900 mV	100 mV
5º	760 mV	0 mV
10º	240 mV	-60 mV
15º	140 mV	-60 mV

Tabla 4.3: Ángulos aceptados entre emisor y receptor dentro de nuestro sistema

Con un ángulo de 20 cm, el receptor no era capaz de captar la señal proveniente

del emisor.

Por último, se han realizado una serie de pruebas con la finalidad de verificar la eficiencia del sistema en diferentes situaciones. Para ello, se ha implementado un script en MATLAB, en el cual se han utilizado las funciones propias para transmitir por el puerto serie. Con la ayuda del programa **SerialPlot** se han volcado los caracteres recibidos en un fichero .csv, teniéndolos así listos para comparar. A continuación se ha usado la función **biterr** de MATLAB, que compara bit a bit una serie de caracteres, indicando cuantos son distintos y el porcentaje de erróneos sobre el total BER (Bit Error Rate). De esta forma se ha conseguido dotar a nuestra verificación del funcionamiento de un mayor rigor. A continuación, se muestra una tabla, en la que se pueden observar las pruebas realizadas para diferentes condiciones de luminosidad y distancia.

Distancia	**Luminosidad**	**Número errores**	**BER**
1 cm	Baja	0	0 %
1 cm	Media	1	2,08 %
1 cm	Alta	8	16,67 %
1,5 cm	Baja	0	0 %
1,5 cm	Media	5	10,42 %
1,5 cm	Alta	9	18,75 %
2 cm	Baja	3	6,25 %
2 cm	Media	7	14,58 %
2 cm	Alta	11	22,92 %
3 cm	Baja	6	12,50 %
3 cm	Media	6	12,50 %
3 cm	Alta	11	22,92 %

Tabla 4.4: BER calculado para distintas situaciones de luminosidad y distancia

En la tabla se pueden observar los dos principales limitantes de nuestro sistema de comunicaciones: la distancia entre emisor-receptor y la luminosidad ambiente. Destacar la situación en la que emisor y receptor se encuentran a 1 cm, pudiendo observar cómo el BER crece desde un 0 % en el mejor de los casos, donde tenemos una luminosidad baja, hasta un 16,67 % para una alta luminosidad.

Como apunte extra a las medidas de la tabla, se ha probado el sistema con una distancia entre emisor-receptor de 10 cm. En esta ocasión, con una luminosidad alta el ruido se superponía completamente a los pulsos, lo que provocaba la detección de más caracteres de los transmitidos.

Capítulo 5

Conclusiones

Durante el desarrollo del Trabajo Fin de Grado se ha experimentado con los sistemas de comunicaciones por luz visible, pudiendo observar tanto las ventajas como desventajas de estos. Se ha conseguido el funcionamiento de uno de ellos con una estructura, desde el punto de vista analógico, sencilla, haciendo especial hincapié en los bloques implicados en la parte digital. Para ello se ha hecho uso de una FPGA Basys 3, en la que mediante el empleo de bloques IP en algunos casos, se ha desarrollado el sistema necesario tanto para transmitir como para recibir caracteres.

Refiriéndonos dentro del emisor a la parte analógica, se ha decidido esa estructura de etapa debido a su sencillez y eficiencia. Mediante el transistor utilizado se logra realizar un control eficiente del encendido y apagado del LED. Con la alimentación seleccionada dentro de este se consigue una iluminación en el diodo más que aceptable a la hora de efectuar la comunicación. Se han realizado pruebas con niveles mayores de voltaje alcanzando una mayor iluminación. Sin embargo, con el fin de unificar las tensiones para todo el sistema, se optó finalmente por una alimentación unificada de ± 7 voltios. Siguiendo en el emisor, aunque ahora centrándonos en la parte digital, destacar la utilización de la modulación OOK codificada sobre Manchester, de la cual se ha demostrado su eficiencia para los sistemas VLC en general y más aún en el nuestro.

Respecto al receptor analógico, resaltar la introducción de un filtro antialiasing para evitar réplicas en la señal. Además se introduce un circuito de control de voltaje, ya que, sin este, podríamos dañar el conversor A/D de la FPGA. Continuando con la parte digital, dentro de la etapa de recepción se han usado varios bloques IP para diferentes funciones, entre otros, el que permite realizar la conversión A/D de los datos recibidos. En cuanto a las tareas para las que se usó un IP, estas fueron tanto la transmisión puerto serie como la recepción, esta última correspondiente al bloque del emisor. El uso de estos IP ha sido posible porque nuestra FPGA forma parte de la

serie 7 de FPGAs de Xilinx, que son las que incluyen dichos bloques. Siguiendo con el conversor A/D, cabe destacar, además, un valor para la frecuencia de muestreo de 77 kSPS, ajustado según la tasa de transmisión de nuestro sistema para conseguir una decisión óptima a través de la media de 2 bits. Por último, resaltar la utilización de un algoritmo, que, a través de una simple suma, decide si lo recibido y muestreado se trataba de un '1' o un '0'.

Para finalizar, se podría englobar todo el sistema desarrollado dentro de los niveles OSI 1 y 2, correspondientes al nivel físico y de enlace. Este modelo se describe en el **Anexo C** de este proyecto. Se han podido observar los rangos de funcionamiento del sistema en diferentes situaciones, como la disminución en la iluminación de la sala, el desfase entre el emisor y el receptor y la introducción de una cierta distancia en el canal entre ambos, finalizando las pruebas con un análisis del BER en alguna de ellas.

5.1. Líneas futuras

Como continuación al proyecto desarrollado, se plantea, en primer lugar, la inclusión de una etapa de alimentación que suponga la sustitución de la fuente utilizada en nuestro sistema. En el **anexo E**, se puede ver un posible diseño que podría encajar perfectamente dentro de nuestro Trabajo Fin de Grado.

Continuando con la parte experimental, se podrían evaluar las prestaciones del sistema respecto a la tasa de transmisión, aumentándola hasta que dejase de funcionar. Para ello los filtros correspondientes al receptor analógico se deberían ajustar a la nueva frecuencia de la señal, de modo que habría que modificar bastante el sistema, sobre todo desde el punto de vista analógico.

Por último y profundizando en la parte digital, se podría implementar alguna modulación usada en este tipo de sistema para conseguir grandes tasas de transmisión, como por ejemplo OFDM, junto con algún tipo de algoritmo de corrección de errores, tal como Reed-Solomon, que mejorase el BER obtenido en nuestro sistema. Además, se podría añadir en el receptor un bloque que realizase un control adaptativo de ganancia de la señal obtenida. Con ello, en situaciones desfavorables como una gran distancia en el canal que separa emisor y receptor, que provocaría la disminución de la amplitud de los pulsos recibidos, podríamos aumentarlos y decidir de una forma más eficaz.

Bibliografía

[1] R. Ji, S. Wang, Q. Liu, and W. Lu, "High-Speed Visible Light Communications: Enabling Technologies and State of the Art," *Applied Sciences*, vol. 8, apr 2018.

[2] I-Cheng Lu, Yen-Liang Liu, and Chih-Han Lai, "High-speed 22 MIMO-OFDM visible light communication employing phosphorescent LED," in *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, pp. 222–224, IEEE, jul 2016.

[3] Yingjun Zhou, Shangyu Liang, Siyuan Chen, X. Huang, and N. Chi, "2.08Gbit/s visible light communication utilizing power exponential pre-equalization," in *2016 25th Wireless and Optical Communication Conference (WOCC)*, pp. 1–3, IEEE, may 2016.

[4] Y.-F. Huang, C.-T. Tsai, H.-Y. Kao, Y.-C. Chi, H.-Y. Wang, T.-T. Shih, and G.-R. Lin, "17.6-Gbps universal filtered multi-carrier encoding of Gan blue LD for visible light communication," 2017.

[5] M. S. Islim, R. Ferreira, X. He, E. Xie, S. Videv, S. Viola, S. Watson, N. Bamiedakis, R. V. Penty, I. White, A. E. Kelly, E. Gu, H. Haas, and M. Dawson, "Towards 10 Gb/s orthogonal frequency division multiplexing-based visible light communication using a Gan violet micro-led," 2017.

[6] H. Chun, S. Rajbhandari, G. Faulkner, D. Tsonev, E. Xie, J. J. D. McKendry, E. Gu, M. D. Dawson, D. C. O'Brien, and H. Haas, "LED Based Wavelength Division Multiplexed 10 Gb/s Visible Light Communications," *Journal of Lightwave Technology*, vol. 34, no. 13, pp. 3047–3052, 2016.

[7] R. X. G. Ferreira, E. Xie, J. J. D. McKendry, S. Rajbhandari, H. Chun, G. Faulkner, S. Watson, A. E. Kelly, E. Gu, R. V. Penty, I. H. White, D. C. O'Brien, and M. D. Dawson, "High Bandwidth GaN-Based Micro-LEDs for Multi-Gb/s Visible Light Communications," vol. 28, no. 19, pp. 2023–2026, 2016.

[8] D. Tsonev, S. Videv, and H. Haas, "Towards a 100 gb/s visible light wireless access network," *Optics Express*, vol. 23, no. 2, p. 1627–1637, 2015.

[9] Y.-C. Chi, D.-H. Hsieh, C.-T. Tsai, H.-Y. Chen, H.-C. Kuo, , and G.-R. Lin, "450-nm gan laser diode enables high-speed visible light communication with 9-gbps qam-ofdm," *Optics Express*, vol. 23, pp. 13051–13059, may 2015.

[10] Y. Wang, X. Huang, J. Zhang, Y. Wang, and N. Chi, "Enhanced performance of visible light communication employing 512-qam n-sc-fde and dd-lms," *Optics Express*, vol. 22, pp. 15328–15334, jun 2014.

[11] H. Li, X. Chen, J. Guo, and H. Chen, "A 550 Mbit/s real-time visible light communication system based on phosphorescent white light LED for practical high-speed lowcomplexity application," vol. 22, pp. 27203–27213, 2014.

[12] H. Parikh, J. Chokshi, N. Gala, and T. Biradar, "Wirelessly transmitting a grayscale image using visible light," 2013.

[13] "Li-Fi revolution: internet connections using light bulbs are 250 times faster than broadband." [En linea]. Disponible en: `https://www.independent.co.uk/news/science/li-fi-revolution-internet-connections\tolerance9999\emergencystretch3em\hfuzz.5\p@\vfuzz\hfuzz-using-light-bulbs-are-250-times-faster-than-broadband-8909320.html`.

[14] H. Lv, L. Feng, A. Yang, P. Guo, H. Huang, and S. Chen, "High Accuracy VLC Indoor Positioning System With Differential Detection," pp. 1–13, 2017.

[15] H. Pranav, "Building a Bidirectional Visible Light Communication,"

[16] B. L. Flores, "Visible light communication," Trabajo Fin de Grado, Universidad Politécnica de Cataluña, 2014.

[17] T. Instruments, "Designing an anti-aliasing filter for ADCs in the frequency domain," 2015.

[18] Xilinx, "7 Series FPGAs and Zynq-7000 SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter User Guide (UG480)," 2018.

[19] Xilinx, "Xilinx PG155 AXI4-Lite IP Interface (IPIF) v2.0, Product Guide for Vivado Design Suite," 2013.

[20] "Welcome to real digital." [En linea]. Disponible en: `https://www.realdigital.org/doc/a9fee931f7a172423e1ba73f66ca4081`.

[21] Xilinx, "AXI UART Lite v2.0 LogiCORE IP Product Guide Vivado Design Suite," 2017.

[22] ARM, "AMBA ® AXI ™ and ACE ™ Protocol Specification AXI3 ™ , AXI4 ™ , and AXI4-Lite ™ ACE and ACE-Lite ™," 2003.

[23] Xilinx, "7 Series FPGAs Clocking Resources User Guide (UG472)," 2018.

[24] A. M. Egea, "Código fuente de visiblelinecommunication." [En linea]. Disponible en: `https://github.com/AMolinaEgea/VisibleLightCommunication`, 2019.

[25] W. Stallings, *Comunicaciones y redes de computadores*. Prentice Hall, 6 ed., 2001.

[26] "Fundamentos de redes," apuntes de clase para 30313, Departamento de Ingeniería Electrónica y Comunicaciones, 2015.

[27] J. Rufo, J. Rabadan, F. Delgado, C. Quintana, and R. Perez-Jimenez, "Experimental Evaluation of Video Transmission Through LED Illumination Devices," p. 1411–1416, 2010.

[28] "Dispositivos y sistemas de transmisión Óptica," apuntes de clase para 30335, Departamento de Ingeniería Electrónica y Comunicaciones, 2017.

Lista de Figuras

Lista de Tablas

Glosario

AAF Antialiasing Filter

ADC Analogic to Digital Converter

BER Bit Error Rate

DAQ Data Acquisition

DRP Dynamic Reconfiguration Port

FPGA Field Programmable Gate Array

GPS Global Positioning System

IP Intellectual Property

LD Laser Diode

LED Light Emitting Diode

LSB Less Significative Bit

MIMO Multiple-Input Multiple-Output

MSB More Significative Bit

OFDM Orthogonal Frequency Division Multiplexing

OOK On-Off Keying

PCB Printed Circuit Board

Pmod Peripherical module

QAM Quadrature Amplitud Modulation

RF Radio Frequency

UART Universal Asynchronous Receiver-Transmitter

UPD Ultrafast Photodiode

VHDL VHSIC Hardware Description Language

VLC Visible Light Communication

VLCC Visible Light Communication Consortium

Anexos

Anexos A

Componentes y dispositivos utilizados en el proyecto

A.1. Lista de componentes

Componente	**Modelo y Datasheet**
Transistor NPN	BC107B http://www.farnell.com/datasheets/1661949.pdf
LED de alta luminosidad	SSL-LX5093UWW http://www.farnell.com/datasheets/2292945.pdf
Fotodiodo	VTB-1013BH http://www.farnell.com/datasheets/1642037.pdf
Amplificador operacional	LM358N http://www.ti.com/lit/ds/snosbt3i/snosbt3i.pdf
Transistor PNP	BC557B http://www.farnell.com/datasheets/296678.pdf
Potenciómetro	T93YB https://www.vishay.com/docs/51026/t93.pdf
Conversor USB-UART	CP2102 https://www.sparkfun.com/datasheets/IC/cp2102.pdf

Tabla A.1: Lista de componentes utilizados a lo largo del proyecto

A.2. Lista de dispositivos

Dispositivos	**Modelo**
Generador de señales	GFG-8216A
Fuente de alimentación	HY3003D-3
Osciloscopio	IDS-1104B
FPGA	Basys 3

Tabla A.2: Lista de dispositivos utilizados a lo largo del proyecto

Anexos B

Creación de un nuevo proyecto para la Basys 3

A la hora de crear un proyecto mediante vivado deberemos seguir una serie de pasos en los que finalmente seleccionaremos la placa donde vamos a realizar nuestro trabajo. Una vez abierto vivado, seleccionaremos la pestaña **File>Project>New**. A continuación nos aparecerá una ventana en la que debemos seleccionar **Next**; en este momento habrá que darle un nombre a nuestro proyecto e indicar dónde lo vamos a crear. Además debemos seleccionar la ventana **create project subdirectory** y, una vez hecho esto, le daremos de nuevo a **Next**. En este momento debemos seleccionar el tipo de proyecto, en nuestro caso seleccionaremos **RTL project** y la pestaña **Do not specify sources at this time**. Luego pulsaremos **Next** y nos aparecerá la última de las pestañas, en la cual deberemos configurar los diferentes campos según la placa que vayamos a utilizar. Los campos que debemos configurar en nuestro caso se muestran a continuación:

- **Category**: General Purpose
- **Family**: Artix-7
- **Package**: cpg-236
- **Speed grade**: -1
- **Part**: xc7a35tcpg236-1

A continuación se muestran una serie de imágenes representativas del proceso mencionado anteriormente.

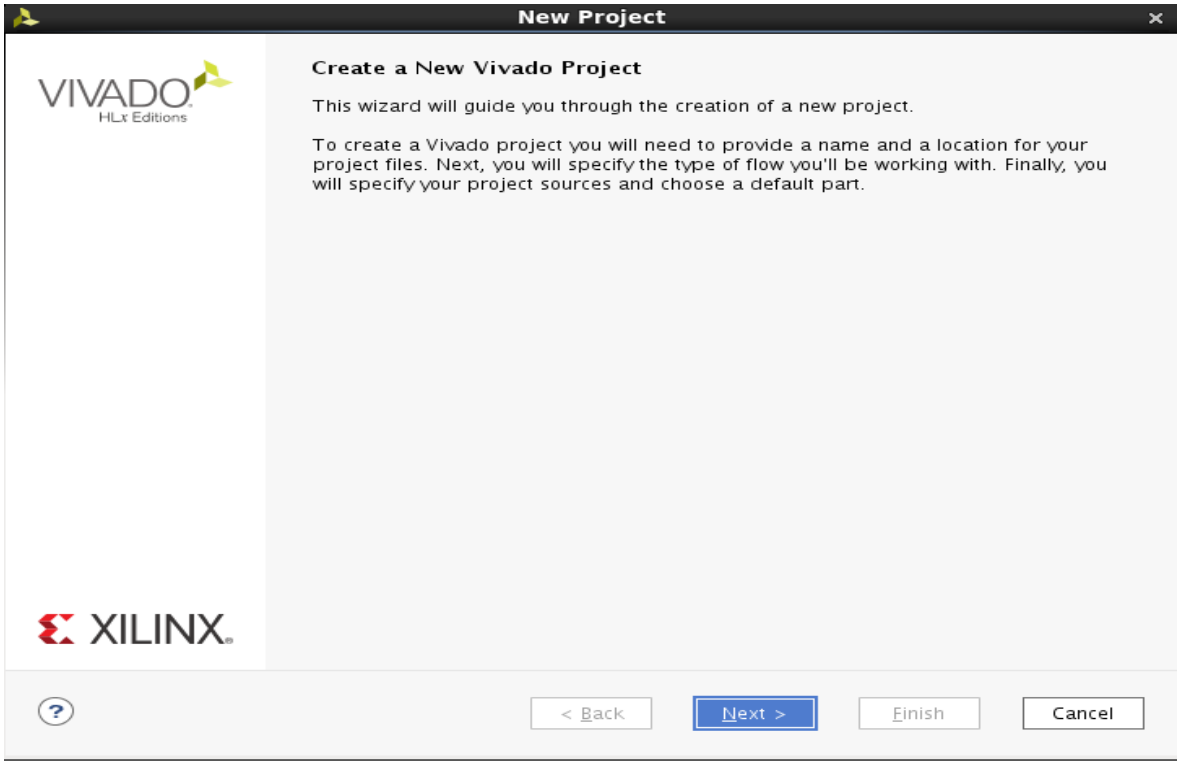

Figura B.1: Ventana de selección de nuestra placa

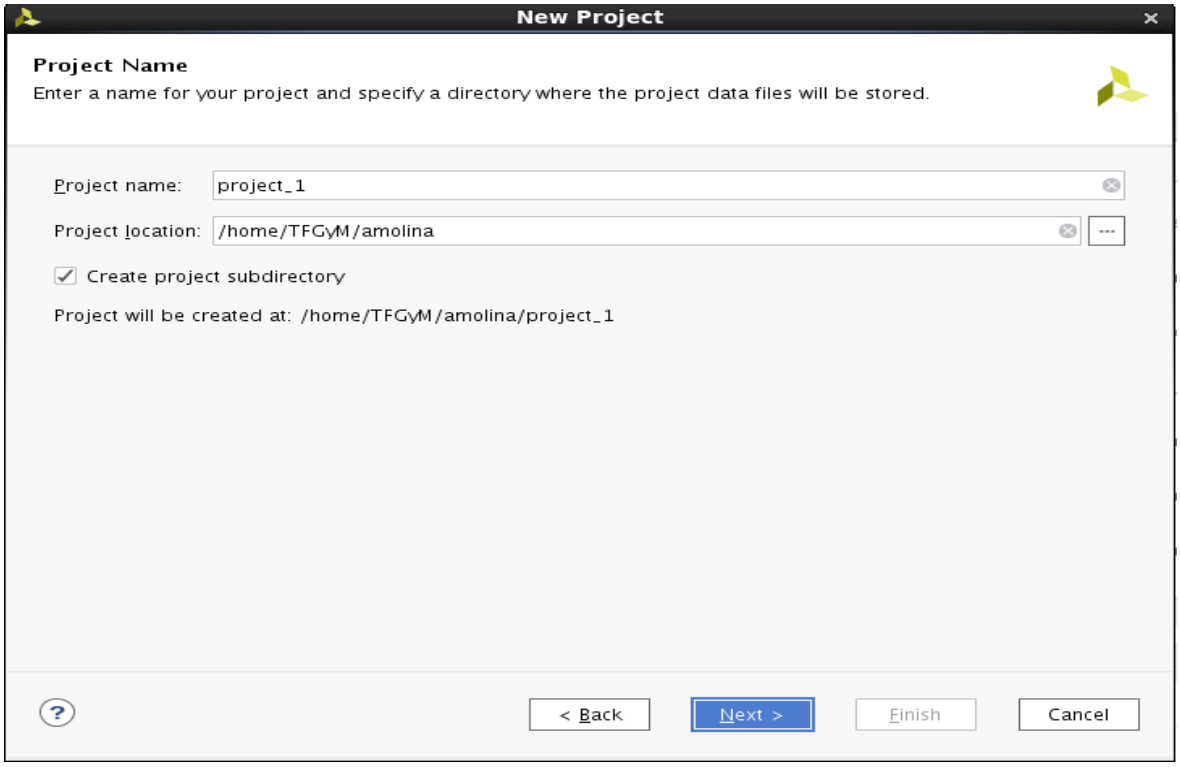

Figura B.2: Ventana de selección de nuestra placa

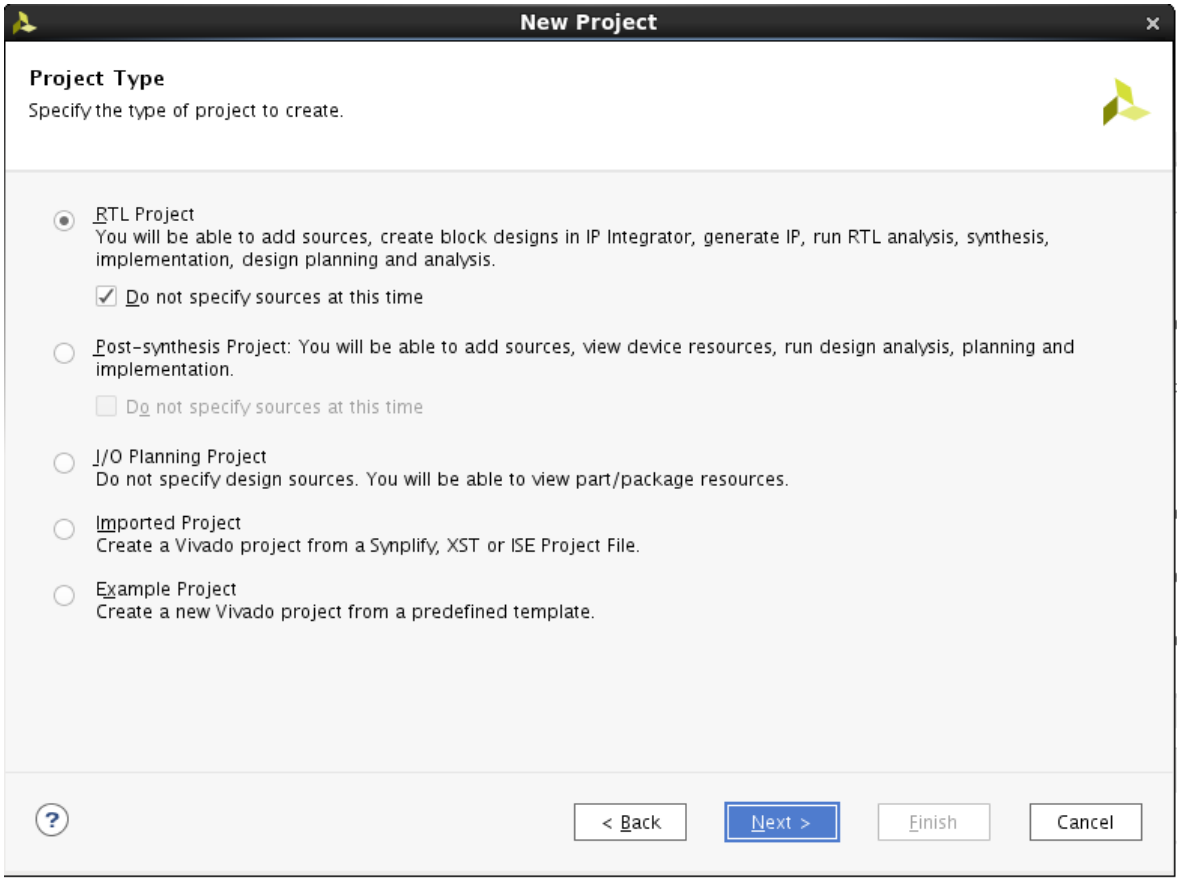

Figura B.3: Ventana de selección de nuestra placa

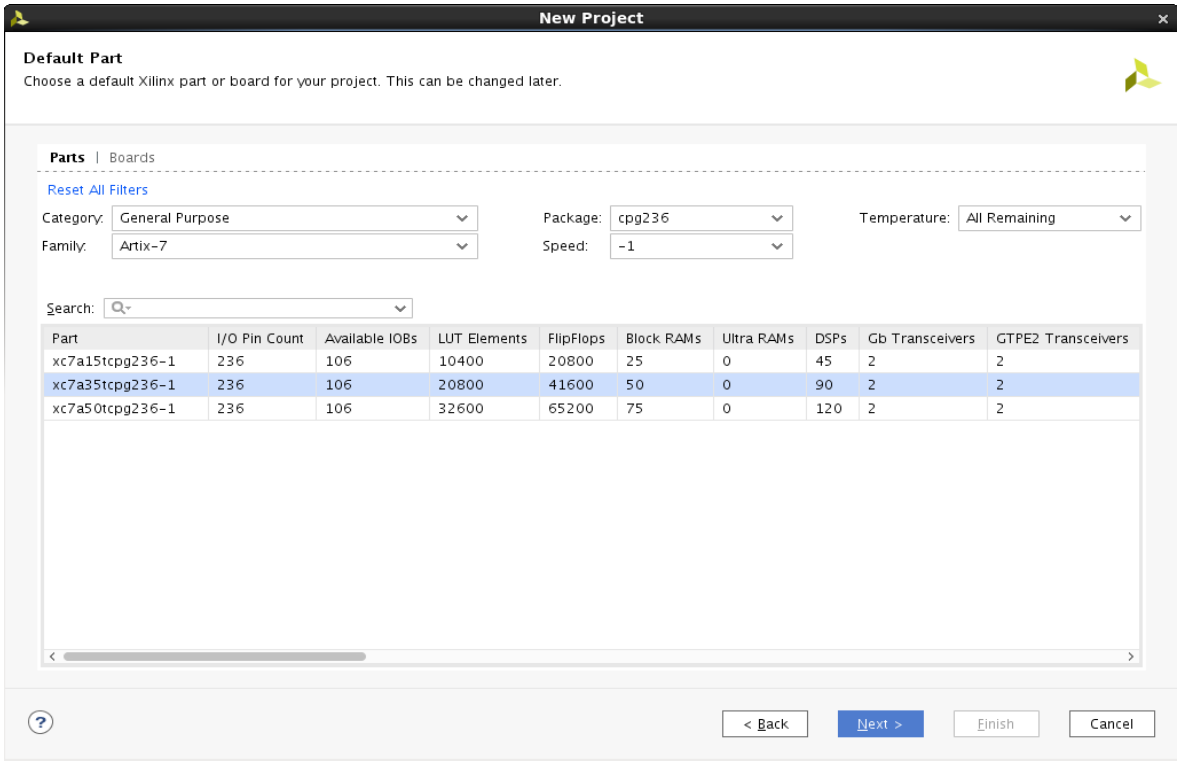

Figura B.4: Ventana de selección de nuestra placa

Anexos C

Modelo OSI

C.1. Introducción

El modelo de interconexión de sistemas abiertos (OSI), se trata de una arquitectura de comunicación entre computadores, desarrollada por la ISO, Organización Internacional de Estandarización [25]. A continuación se muestra una figura en la que aparecen las 7 capas en las que se divide el modelo. Además se incluye una breve explicación de cada una de ellas.

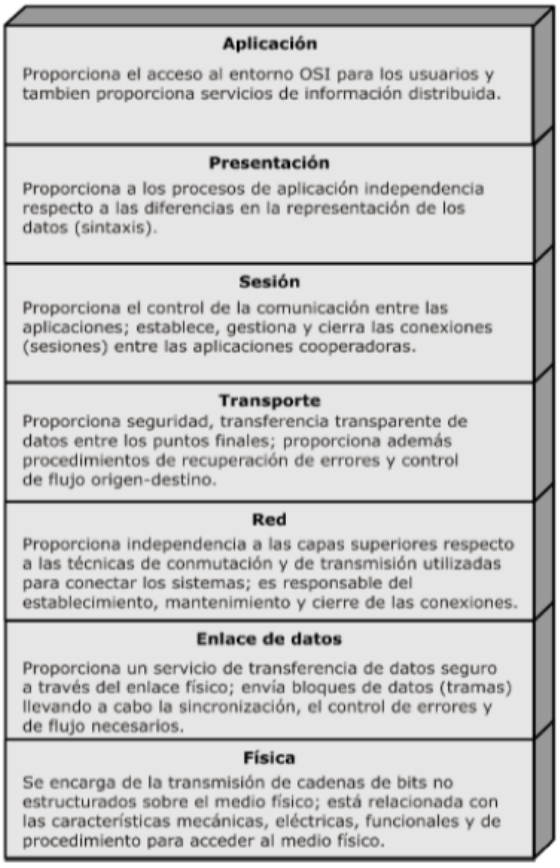

Figura C.1: Niveles dentro del modelo OSI [25]

C.2. Nivel físico

Se trata de la más básica de las 7. Define las reglas usadas durante la transmisión de los bits. Dentro de ella se deberán definir características como el medio de transmisión, forma en la que se representan los bits, velocidad a la cual son transmitidos, voltajes usados, tipo de codificación, modulación de la señal, etc.

C.3. Nivel de enlace

Su función principal es dotar al enlace físico de seguridad, se encargará de realizar el sincronismo de la trama junto con la detección y corrección de errores. En redes de difusión, realiza también el control de acceso al medio compartido, mientras que en las de conmutación se encarga de gestionar el establecimiento, mantenimiento y liberación de los enlaces [26].

C.4. Nivel de red

Se ocupa de realizar la transferencia de información entre sistemas cuando estos no tienen acceso directo entre ellos. En el caso de tener enlaces directos punto-a-punto no se necesita esta capa, ya que la de enlace puede realizar las funciones básicas de gestión.

C.5. Nivel de transporte

Se encarga del intercambio de los datos entre un emisor y un receptor. El servicio de transporte puede ser de dos tipos: no orientado a conexión u orientado a conexión. Este último garantiza la desaparición de errores, la recepción ordenada de los datos y la ausencia de pérdidas.

C.6. Nivel de sesión

En una importante cantidad de casos sus servicios son prescindibles. Sin embargo, entre sus funciones, puede actuar de interfaz entre el usuario y la red, establecer la forma en que dos procesos se intercambian los datos, restablecer la sesión en caso de corte de conexión, proporcionar el control de la comunicación entre aplicaciones, etc.

C.7. Nivel de presentación

Define el formato de los datos intercambiados entre aplicaciones [25], evitando así problemas con, por ejemplo, diferencias en la forma de codificación de los bits.

C.8. Nivel de aplicación

Se encarga de dar al usuario el acceso al modelo OSI. Algunos ejemplos conocidos de protocolos de la capa de aplicación son:

- **HTTP**: Protocolo utilizado por clientes y servidores de WWW para comnuicarse entre sí.
- **TELNET**: Permite realizar una conexión de forma remota de un ordenador a otro.
- **FTP**: Posibilita la copia de ficheros de un ordenador a otro.
- **SMTP**: Servicio de correo a través de servidores.

Anexos D

Circuitos finales desarrollados en PCB

A continuación se muestran las imágenes correspondientes a los circuitos finales desarrollados en nuestro sistema VLC.

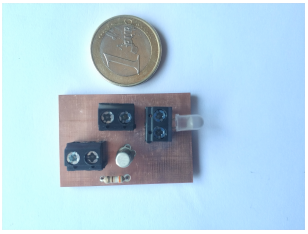

Figura D.1: Imagen de la PCB correspondiente al emisor del sistema

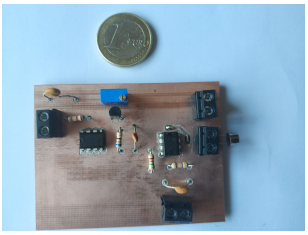

Figura D.2: Imagen de la PCB correspondiente al receptor del sistema

Anexos E

Etapa de alimentación alternativa

A continuación se muestra una figura en la que se representa una posible solución a la sustitución de la fuente como etapa de alimentación de nuestro sistema VLC.

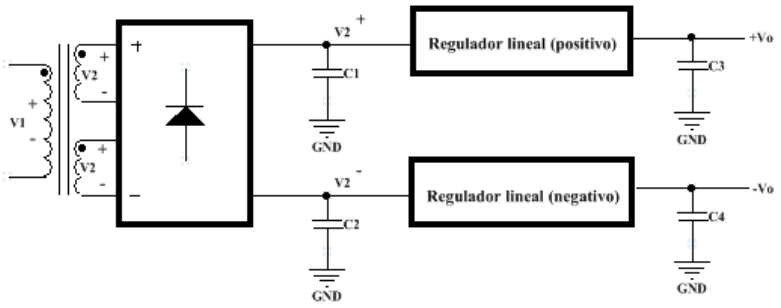

Figura E.1: Posible circuito de alimentación de nuestro sistema

En ella se puede observar cómo mediante un transformador de doble devanado pasamos de una tensión $\mathbf{v_1}$ a $\mathbf{v_2}$. La tensión en $\mathbf{v_1}$ será la de la red eléctrica, 220 v, 50 Hz, mientras que la de $\mathbf{v_2}$ dependerá del transformador utilizado. La señal obtenida se introducirá a un rectificador de onda completa obteniendo $\mathbf{v_2}^+$ y $\mathbf{v_2}^-$, las cuales son la misma tensión cambiada de signo. Ambas se introducirán por los correspondientes reguladores lineales para acabar obteniendo $+\mathbf{v_0}$ y $-\mathbf{v_0}$ respectivamente, que deberán ser ± 7 voltios, tensiones de alimentación usadas en nuestro sistema. Para ello habría que buscar los componentes adecuados. Respecto a los condensadores que aparecen en la figura E.1, destacar su función, la cual es conseguir una mejora en la respuesta transitoria del regulador. El valor de $\mathbf{C_1}$ y $\mathbf{C_2}$ será el mismo, ocurriendo de igual forma para $\mathbf{C_3}$ y $\mathbf{C_4}$, siendo entre ellos distinto.

Anexos F

Formas de generación de luz blanca

La generación de luz blanca mediante LEDs se puede realizar de dos formas, las cuales tienen ventajas y desventajas. Estas son:

- Utilización de LEDs de color rojo, verde y azul, cuya mezcla producirá luz blanca.
- Utilización de un único LED, el cual será en la mayoría de los casos de color azul, con una capa de fósforo amarilla.

Respecto a las ventajas y desventajas de estos, destacar en el LED que utiliza el fósforo un menor consumo respecto al otro, un bajo coste y una alta eficiencia, siendo una gran desventaja el bajo ancho de banda debido a la baja respuesta temporal de este material [27]. Por tanto, la utilización de uno u otro dependerá de la aplicación que se le vaya a dar, siendo el de 3 emisores una buena opción en aplicaciones de alta velocidad a causa de su mayor ancho de banda.

En nuestro caso se ha hecho uso del segundo tipo de LED enumerado principalmente por su coste, falta de necesidad de un gran ancho de banda, y facilidad de uso.

Anexos G

Tipos de fotodiodos y modos de operación

G.1. Tipos de fotodiodos

Los principales fotodiodos que se están usando en la actualidad son el PIN y el de avalancha (APD). A continuación, sin entrar en profundidad dentro de cada uno, se compararán uno y otro con el fin de justificar nuestra elección dentro de este Trabajo Fin de Grado.

La principal diferencia entre un fotodiodo PIN y un APD, es que este último multiplica internamente la corriente recibida antes de entrar en el circuito amplificador [28]. Esto implica que la responsividad de este venga multiplicada en un factor M frente a la de un PIN. Por otro lado, no todo son ventajas, ya que entre otras cosas, el tiempo de subida, tiempo en que la corriente pasa del 10 % al 90 % de su valor directo nominal, es mayor en un APD que en un PIN. Otras desventajas del APD frente al PIN son: un mayor ruido, un menor ancho de banda y un precio notablemente más alto. Estas, junto con la necesidad de un valor muy alto de voltaje para polarizarlo, hicieron que en nuestro Trabajo Fin de Grado nos decantásemos por un fotodiodo tipo PIN.

G.2. Modos de operación

A continuación se enumeran los diferentes modos en los que pueden trabajar los fotodiodos.

- **Circuito abierto o fotovoltaico**
- **Cortocircuito**
- **Polarización inversa o fotoconductivo**

Normalmente se recomienda el trabajo del fotodiodo en modo fotoconductivo, tal como se ha realizado en nuestro caso, debido a entre otras cosas una mejor respuesta en frecuencia [28]. Destacar, por último, el intento en nuestro caso de configuración del fotodiodo en modo fotovoltaico con el objetivo de eliminar la alimentación negativa de nuestro sistema, no consiguiendo la recepción de la señal, por lo que se terminó optando por la configuración mencionada, modo fotoconductivo.